

Bill McCarty

PHP 4

Traducere de Mihai Mănăstireanu



130178
B.C.U. - IASI

Teora

Titlul original: **PHP 4 A BEGINNER'S GUIDE**

• Copyright © 2002 Teora

Toate drepturile asupra versiunii în limba română aparțin Editurii **Teora**.
Reproducerea integrală sau parțială a textului sau a ilustrațiilor din această carte
este posibilă numai cu acordul prealabil scris al Editurii **Teora**.

Original edition copyright © 2001 by The McGraw-Hill Companies. All rights reserved.
Romanian edition copyright © 2001 by Editura Teora. All rights reserved.

Teora

Calea Moșilor nr. 211, sector 2, București
fax: 01/210.38.28
e-mail: teora@teora.kappa.ro

Teora – Cartea prin poștă

CP 79-30, cod 72450 București, România
tel: 01/252.14.31
e-mail: cpp@teora.kappa.ro

Copertă: Gheorghe Popescu

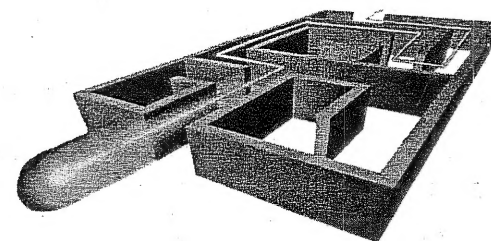
Tehnoredactare: Techno Media

Director Editorial: Diana Rotaru

Președinte: Teodor Răducanu

NOT 4965 CAL PHP4
ISBN 973-601-737-0

Printed in Romania



Cuprins

Introducere 9

PARTEA I

Scrierea programelor PHP elementare

1 Crearea programelor PHP	13
2 Elementele constructive ale limbajului PHP	25
3 Crearea formularelor HTML	38
4 Accesul la date	61
5 Lucrul cu valori scalare	71

PARTEA A II-A

Scrierea unor programe PHP cu un grad avansat de complexitate

6 Scrierea instrucțiunilor PHP condiționale	85
7 Utilizarea funcțiilor	105
8 Utilizarea tablourilor	123
9 Utilizarea șirurilor	139

PARTEA A III-A

Lucrul cu date stocate

10 Utilizarea variabilelor cookie	159
11 Lucrul cu fișiere și cataloage	169
12 Expedierea și recepționarea mesajelor de poștă electronică	209
13 Noțiuni fundamentale despre bazele de date și SQL	234

PARTEA A IV-A

Utilizarea funcționalităților avansate ale limbajului PHP

14	Accesul la bazele de date relaționale	269
15	Utilizarea claselor și a obiectelor	302
16	Utilizarea șabloanelor de aplicație	318
17	Depanarea scripturilor PHP	327

PARTEA A V-A

Anexe

A	Răspunsuri la testele de verificare	345
B	Instalarea PHP	356
C	Resurse PHP	362
D	Elemente fundamentale ale sistemului de operare UNIX	364
E	Caractere escape	383
F	ASCII	384
G	Operatori PHP	387
H	Securitate	389
I	Funcții PHP	394
	Index	423

*Această carte este dedicată lui John C. Reynolds,
Chief Information Officer la Azusa Pacific University.
Fie ca Dumnezeu să-i binecuvânteze eforturile de a conduce
APU la o nouă percepție asupra tehnologiei și a beneficiilor
potențiale pe care aceasta le oferă.*

Despre autor

Bill McCarty a scris peste zece cărți de programare a calculatoarelor, abordând subiecte precum Java, prelucrarea distribuită a datelor, C++ și Linux, și este redactor colaborator la revista *Linux Magazine*, pentru care scrie un editorial lunar. Este profesor asociat de Web și tehnologia informației la Facultatea de afaceri și management a Universității Azusa Pacific, Azusa, California, unde a predat vreme de șapte ani în cadrul programului de informatică aplicată al facultății respective. Bill a dobândit titlul de doctor în gestiunea sistemelor informaționale la Universitatea Claremont, California, și a lucrat vreme de 15 ani ca programator și gestionar de programe. În prezent, locuiește în La Habra, California, alături de familia sa.

Din partea autorului

Mulți oameni au colaborat la această carte. Margot Maley de la Waterside Productions, Inc. l-a prezentat pe autor la editura Osborne/McGraw-Hill. Rebekah Young a contribuit la conturarea structurii noii cărți. Paulina Pobocha și Monica Faltiss s-au ocupat atât de autor, cât și de cartea aflată la început, cu încurajări sau îndemnuri, după cum o cereau circumstanțele. John Steele a descoperit multe erori de ordin tehnic și a furnizat numeroase sugestii de valoare. Cu puțin noroc, au rămas erori foarte puține sau chiar deloc; cu toate acestea, toate erorile care apar sunt numai din vina autorului, care ar fi trebuit să facă mai puține greșeli la început decât să le lase pe toate, ca un leneș, în seama lui John. Mike McGee a „curățat” cuvinte, fraze și propoziții incorecte, neclare sau... mai rău. Mike și-a câștigat de mai multe ori salariul. Familia autorului – Jennifer, Patrick și Sara – au păstrat gospodăria pe linia de plutire pe durata lungilor luni necesare pentru scrierea cărții, succesul acesteia datorându-se atât eforturilor lor, cât și celor ale autorului. În final, ocrotitorul autorului, Iisus Christos, a fost și rămâne unicul reper autentic de adevăr și valoare al autorului, constituind o sursă de inspirație a muncii acestuia.

Introducere

PHP este una dintre cele mai interesante tehnologii existente în prezent. Deoarece îmbină caracteristici dintre cele mai complexe cu simplitatea în utilizare, PHP a devenit rapid un instrument de frunte pentru dezvoltarea aplicațiilor în Web. Totuși, spre deosebire de alte instrumente populare pentru dezvoltarea aplicațiilor Web, cum este Perl, PHP este un limbaj de programare comod pentru începători, chiar și pentru cei care nu au mai desfășurat activități de programare în trecut.

Dacă sunteți un cunoscător al limbajului HTML, dar nu aveți experiență în materie de programare, vă puteți pune întrebarea: care sunt funcționalitățile suplimentare pe care le poate asigura cunoașterea limbajului PHP? Ca și alte limbaje de scripting pentru Web, PHP vă permite să furnizați un *conținut Web dinamic*, adică un conținut Web care se modifică automat de la o zi la alta sau chiar de la un minut la altul. Conținutul Web este un element important în susținerea traficului unui sit Web; de regulă, vizitatorii nu vor mai reveni la o pagină Web care conține aceleași informații ca și cele prezentate la ultima vizită. Pe de altă parte, siturile Web frecvent actualizate *pot* atrage cantități enorme de trafic.

Mai mult, spre deosebire de limbajele de scripting, precum JavaScript, PHP rulează pe serverul Web, nu în browserul Web. În consecință, PHP poate obține accesul la fișiere, baze de date și alte resurse inaccesibile programului JavaScript. Acestea constituie bogate surse de conținut dinamic, care atrag vizitatorii.

Această carte este menită a prezenta cititorului elemente introductive de programare și dezvoltare în Web folosind PHP. Este important să rețineți că volumul de față nu reprezintă decât un punct de plecare. Dezvoltarea Web este o activitate solicitantă, iar viitorul dezvoltator Web trebuie să dispună de multe abilități, printre care și pe aceea de programator. Sunt încrezător că dezvoltatorii Web începători vor găsi în această carte un prim pas util și amical în activitatea de programare în PHP. Mai mult, cartea a fost atent concepută cu scopul de a asigura un fundament pentru învățarea noțiunilor ulterioare pe care le implică o bună cunoaștere a limbajului PHP. După studiul materialului prezentat aici, cititorul trebuie să fie pregătit a învăța mai multe despre caracteristici și funcționalități PHP mai complexe, precum XML, LDAP și cele legate de comerțul electronic.

Modul de organizare a cărții

Această carte începe cu... începutul, explicând modul de funcționare a limbajului PHP, apoi trece la detalierea modului de creare a programelor PHP, scoțând în evidență detaliile mici, dar importante, precum modul de încărcare a scripturilor PHP. Volumul conține o trecere în revistă a elementelor fundamentale ale limbajului HTML și o explicare a formularelor HTML. Cele șaptesprezece module ale cărții

abordează în mod controlat o prezentare gradată a conceptelor folosite în activitatea de programare și a elementelor specifice limbajului PHP, fiecare modul conținând elemente destinate a reîmprospăta memoria și înțelegerea cititorului. De asemenea, cartea conține un modul (Modulul 13) care explică elementele fundamentale ale bazelor de date relaționale.

Scopuri

Fiecare modul începe cu un set de scopuri explicite, astfel încât dumneavoastră să aveți o idee privind integrarea fiecărui modul în imaginea de ansamblu.

Test de evaluare

Un auto-test, care vă ajută să vă evaluați nivelul de progres, apare de asemenea în cadrul fiecărui modul. Răspunsurile la aceste teste de evaluare se pot găsi în Anexa A.

Teste „la minut”

Fiecare secțiune principală a cărții conține un test „la minut”, un auto-test care vă ajută să nu adormiți la volan. Răspunsurile la aceste teste se pot găsi la baza paginii unde se află testele respective.

Sfatul specialistului

Cartea include multe casete de acest tip. Textele incluse în casete extind materialul prezentat în capitolul asociat și sporesc calitatea acestuia. Deseori, casetele „Sfatul specialistului” conțin materiale mai avansate, care nu sunt importante pentru programatorul PHP începător, dar ajută cititorul să examineze în perspectivă problemele de la nivelul intermediar al limbajului PHP. Casetele „Sfatul specialistului” folosesc un format de tip întrebare-răspuns.

Proiecte

Fiecare modul conține unul sau mai multe proiecte care vă indică modul de aplicare a conceptelor și tehnicilor explicate în modulul respectiv. Apoi, veți putea folosi aceste proiecte drept bază pentru studiu și experimente ulterioare. Deseori, proiectul furnizează un bun punct de plecare pentru propriul dumneavoastră program practic.

Nu este necesară experiența anterioară în domeniul programării

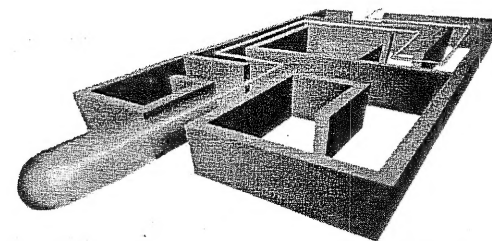
În această carte, se presupune că sunteți într-o oarecare măsură familiarizat cu HTML... și cam atât. Mai ales, nu se pornește de la premisa că aveți experiență în domeniul programării. De aceea, este explicat modul de creare a formularelor HTML, modul de programare și utilizare a limbajului PHP și sunt descrise toate elementele necesare pentru a dezvolta situri Web simple, susținute de baze de date, folosind PHP.

Programe necesare

Pentru a rula exemplele și proiectele din această carte, veți avea nevoie de acces la un server PHP care rulează PHP versiunea 4. Puteți instala limbajul PHP pe propriul dumneavoastră calculator, folosind informațiile furnizate în Anexa B. Totuși, ca începător, este mai convenabil să folosiți un server Web administrat de o altă persoană. Anexa C vă indică unele situri Web care identifică furnizorii de servicii Internet (ISP) care acceptă PHP. Mulți furnizori de servicii Internet asigură PHP la prețuri de 20 USD lunar (în S.U.A. – N. T.) sau chiar mai puțin.

Nu uitați de programele din Web

Codul sursă aferent tuturor exemplelor și proiectelor din această carte este disponibil gratuit în Web, la adresa <http://www.osborne.com>.



Partea I:

Scrierea programelor PHP elementare

Modulul 1

Crearea programelor PHP

Scopuri

- Învățați să creați un script PHP
- Învățați să scrieți instrucțiuni PHP care trimit text la un browser Web
- Învățați să documentați un script PHP
- Învățați să încărcați un script PHP într-un server prin intermediul protocolului FTP
- Învățați să executați un script PHP

În acest modul veți învăța modul de creare și de executare a programelor PHP. Dacă PHP nu este instalat în sistemul dumneavoastră, nu sunteți pregătit pentru a rula programele PHP demonstrative prezentate în acest modul. Înainte de a rula programele prezentate, trebuie să instalați și să testați limbajul PHP respectând instrucțiunile date în Anexa B.

Crearea unui script PHP

Un script PHP poate fi foarte simplu sau foarte complex. Totuși, crearea chiar și a unui script PHP complex este extrem de simplă, necesitând numai un editor de texte obișnuit. În această secțiune, veți învăța să creați scripturi PHP simple, care

trimit unui browser Web date de ieșire sub formă de text. De asemenea, veți învăța să vă documentați scripturile, astfel încât dumneavoastră și alte persoane să puteți înțelege rapid scopul și structura acestora.

Scrierea scripturilor PHP

Pentru a crea scripturi PHP, majoritatea programatorilor PHP folosesc un editor de texte obișnuit. Puteți folosi orice editor de texte doriți. Sub Microsoft Windows, programatorii PHP folosesc frecvent programul Windows Notepad. Dacă preferați, puteți folosi WordPad sau chiar un procesor de texte, precum Microsoft Word. Totuși, dacă folosiți un instrument diferit de Notepad, trebuie să luați măsuri pentru a salva scriptul dumneavoastră sub formă de document text; în caz contrar, fișierul script conține informații de formatare care vor deruta serverul PHP.

Dacă folosiți UNIX sau Linux, puteți crea scripturi PHP folosind un program precum vi, emacs sau pico. Programul în sine nu contează, atâta vreme cât poate crea fișiere text ASCII.

Scrierea scheletului programelor PHP

Fiecare program PHP include două linii speciale, care indică serverului PHP că textul cuprins între cele două linii este alcătuit din instrucțiuni PHP. Practic, aceste linii pot fi asimilate copertelor unei cărți, care păstrează unitatea programului dumneavoastră PHP.

Pentru a începe să scrieți un program PHP, lansați editorul dumneavoastră de texte preferat și introduceți următoarele două linii în spațiul de lucru al editorului:

```
<?php
?>
```

? Sfatul specialistului

Întrebare: Când scriu programe în C, folosesc un mediu integrat de dezvoltare (IDE*) precum Microsoft Visual C++, care reprezintă o gazdă a unor caracteristici speciale, care simplifică proiectarea, codificarea și testarea programelor. Există medii de tip IDE pentru PHP?

Răspuns: Unele editoare de texte, precum vi, asigură un suport special pentru scrierea programelor PHP. De exemplu, caracteristica de colorare a elementelor de sintaxă, prezentă în vi, determină scrierea diferitelor elemente ale programelor PHP în culori diferite. Mulți programatori PHP sunt de părere că procedeul de colorare a elementelor de sintaxă facilitează depistarea erorilor din programele proprii.

* Abreviere de la Integrated Development Environment. – N.T.

Unele editoare de texte, precum HomeSite al firmei Allaire, asigură colorarea elementelor de sintaxă și alte caracteristici care vin în sprijinul programatorilor PHP, precum manualele on-line și constructorii de expresii. Cu toate acestea, când învățați să scrieți programe PHP, probabil că veți găsi utilizarea unui editor de texte obișnuit mai simplă decât folosirea unui editor echipat cu funcționalități PHP speciale. În caz contrar, o bună parte din timpul dumneavoastră va fi alocată învățării modului de utilizare a instrumentului respectiv, nu scrierii programelor PHP în sine. După ce veți căpăta experiență în scrierea programelor PHP, trebuie să examinați instrumente care vă pot ajuta în activitatea dumneavoastră. În acel moment, consultați lista editoarelor PHP, disponibilă în Web la adresa <http://www.itworks.demon.co.uk/phpeditors.htm>.

Apoi, salvați scriptul dumneavoastră elementar sub formă de fișier text, cu un nume care respectă următoarele reguli:

- Numele fișierului trebuie să fie alcătuit numai din caractere minuscule, cifre și liniuțe. Utilizarea spațiilor și a altor caractere este interzisă.
- Extensia numelui fișierelor trebuie să fie .php.

Asigurați-vă că ați ales un nume semnificativ, care să descrie funcția scriptului dumneavoastră, astfel încât să-l puteți identifica rapid după săptămâni sau chiar luni de la crearea acestuia. Veți descoperi că liniuțele sunt utile pentru separarea cuvintelor care alcătuiesc numele fișierului, mărind astfel lizibilitatea acestuia. De exemplu, un fișier care conține un script PHP ce vă permite să vizualizați nivelurile stocurilor aflate pe inventar poate primi numele `niveluri-stoc.php`. Chiar și la mult timp după crearea fișierului respectiv, nu veți avea probleme în a determina scopul acestuia.

? Sfatul specialistului

Întrebare: Aceste reguli de denumire a fișierelor par a avea un caracter deosebit de limitativ. Nu pot folosi și alte caractere pentru denumirea fișierelor care conțin scripturi PHP?

Răspuns: Ba da, puteți. Dar utilizarea altor caractere vă poate provoca necazuri. De exemplu, numele de fișiere din Microsoft Windows nu sunt sensibile la diferența între majuscule și minuscule, în timp ce numele de fișiere din UNIX prezintă această sensibilitate. De asemenea, majoritatea sistemelor de operare prescriu reguli de denumire a fișierelor care diferă de regulile pe care trebuie să le respecte adresele Web (URL). Puteți evita problemele care apar datorită acestor diferențe folosind numai litere minuscule, cifre și liniuțe în numele fișierelor care conțin scripturi PHP.



Test „la minut”

- Care dintre următoarele nume de fișiere respectă regulile date pentru denumirea fișierelor care conțin scripturi PHP?
 - ScriptulMeu.php
 - scriptul_tău.php
 - scriptul-lui.php3
 - scriptul-ei.php

Crearea datelor de ieșire pentru un browser Web

Programele PHP execută trei categorii de operații elementare:

- Obțin date de la un utilizator.
- Execută prelucrări ale datelor, respectiv obțin accesul la datele stocate în fișiere și baze de date și le manipulează.
- Afișează date astfel încât un utilizator să le poată vizualiza.

Primele două operații sunt oarecum mai dificil de realizat decât cea de-a treia. Totuși, afișarea datelor astfel încât acestea să fie vizibile utilizatorului este o operație foarte simplă.

Așa cum paragrafele unui text scris sunt compuse din propoziții, programele PHP sunt alcătuite din instrucțiuni. Regulile care controlează formarea propozițiilor scrise în limba engleză se numesc *sintaxă*.* Același termen este folosit și pentru a desemna regulile care guvernează formarea instrucțiunilor PHP.

Iată o „rețetă” sintactică pentru crearea instrucțiunii PHP care trimite date de ieșire la un browser Web, astfel încât acestea să fie vizibile pentru un utilizator. Această instrucțiune se numește *instrucțiune de reflectare***.

```
echo("scrieti aici un text oarecare");
```

Observați că instrucțiunea începe de la cuvântul `echo` și se încheie cu un caracter punct și virgulă. Parantezele și ghilimelele duble se folosesc pentru delimitarea unei expresii de tip text, în cazul nostru *scrieti aici un text oarecare*, care apare la mijlocul instrucțiunii. Așa cum este indicat prin caracterele scrise cursiv, în locul propoziției *scrieti aici un text oarecare* puteți plasa aproape orice text. Totuși, pentru moment, trebuie să includeți numai litere, cifre, spații și semne de punctuație folosite în

Răspunsuri la test:

- Nu; conține litere scrise cu majuscule
- Nu; conține liniuțe de subliniere
- Nu; extensia fișierului trebuie să fie `.php`
- Da

* Definiția sintaxei este valabilă pentru *orice* limbă, nu numai pentru limba engleză – N.T.

** În original *echo statement* – N.T.

alfabetul latin, precum virgula, caracterul punct și virgulă, punctul, semnul de întrebare și semnul exclamării. De asemenea, puteți include caracterele `<` `>`, folosite pentru delimitarea etichetelor HTML, respectiv caracterul `/`, folosit pentru a indica membrul de închidere al unei perechi de etichete HTML.

De exemplu, iată o instrucțiune PHP care are drept date de ieșire un fragment dintr-un vers din *Scrisoarea a III-a* de Eminescu:

```
echo("<H2>Iata vine-un sol de pace ...</H2>");
```

Perechea de etichete `H2` determină formatarea datelor de ieșire ca titlu HTML de nivel 2.



Sfatul specialistului

Întrebare: Nu există nici o posibilitate de a include caractere speciale (cum ar fi caracterul ghilimele duble) într-o instrucțiune de reflectare?

Răspuns: Dacă includeți caractere speciale în textul pe care îl folosiți efectiv, puteți avea probleme. De exemplu, dacă încercați să includeți în text un caracter de tip ghilimele duble, veți deruta serverul PHP, deoarece acesta se așteaptă ca ghilimelele duble să apară numai la începutul, respectiv la sfârșitul textului, nu și în interiorul textului. PHP furnizează modalități speciale de evitare a acestei probleme; veți învăța despre ele în Modulul 2.



Test „la minut”

- Scrieți o instrucțiune de reflectare care să scrie numele limbajului dumneavoastră de programare preferat.
- Scrieți o instrucțiune de reflectare care să scrie numele dumneavoastră.

Documentarea unui script PHP

În afară de a furniza nume descriptive fișierelor care conțin scripturile dumneavoastră PHP, trebuie să includeți în fiecare script atât comentarii care să permită unui cititor să determine cu ușurință utilitatea scriptului, cât și alte informații referitoare la script. De exemplu, puteți include un comentariu care precizează numele

* Text adaptat. În original se face trimitere la un eveniment din istoria Statelor Unite, puțin relevant pentru cititorul român. – N.T.

Răspunsuri la test:

- `echo("PHP")` sau similar
- `echo("Bill McCarty")` sau similar

fișierului care conține scriptul, astfel încât acesta să apară în versiunile tipărite ale scriptului.

Iată un model sintactic pentru comentariile PHP:

```
// Scrieți aici comentariul dumneavoastră
```

După cum se poate vedea, un comentariu începe cu două caractere slash, urmate de un spațiu. În continuare, linia conține comentariul dumneavoastră, care poate include orice caractere doriți, inclusiv caractere speciale.

Iată un exemplu simplu de script PHP care include comentarii:

```
<?php
// script-exemplu.php
// Acest script afiseaza un mesaj vizibil pentru utilizator.
echo("Acesta este un script foarte simplu.");
?>
```



Sfatul specialistului

Întrebare: Ce se întâmplă dacă doresc să creez un comentariu PHP care se extinde pe mai multe linii? Cum trebuie să procedez?

Răspuns: O modalitate de a crea un comentariu PHP pe mai multe linii este de a începe fiecare linie cu ajutorul caracterelor //. Totuși, puteți crea un comentariu din mai multe linii și în alte moduri, dacă preferați. Iată un exemplu:

```
/*
Acesta este un comentariu pe mai multe linii. Poate fi alcatuit dintr-un numar
oricat de mare de linii.
*/
```

Pentru a începe un comentariu alcătuit din mai multe linii, scrieți caracterele /*, iar pentru a încheia comentariul, scrieți caracterele */. Între cele două perechi de caractere, puteți scrie orice text doriți, folosind oricâte linii doriți.



Test „la minut”

- Scrieți un comentariu PHP care conține numele dumneavoastră.
- Scrieți un comentariu PHP pe mai multe linii, care conține adresa dumneavoastră.

Răspunsuri la test:

- // Bill McCarty
- /*
Strada X nr. 123
Oras Y, PA 12345
*/

Executarea unui script PHP

După ce ați creat un script PHP, veți dori să-l executați. Dacă nu v-ați creat scriptul PHP pe un server unde este instalat PHP, mai întâi trebuie să vă încărcați scriptul într-un server. În această secțiune, veți învăța să încărcați și să executați scripturile PHP.

Încărcarea unui script PHP

Probabil că veți avea nevoie de ajutor la încărcarea unui script PHP, deoarece modul în care veți proceda depinde de metoda de obținere a accesului la server și de modalitatea în care administratorul de sistem a configurat serverul. Trebuie să luați legătura cu administratorul de sistem al serverului dumneavoastră și să aflați care este modul de încărcare a scriptului dumneavoastră. Pentru a vă ajuta să înțelegeți răspunsul administratorului de sistem, această sub-secțiune descrie unele situații comune care apar la încărcarea scripturilor.

Dacă obțineți accesul de la distanță la un server Linux sau UNIX prin intermediul protocoalelor Telnet sau SSH, nici măcar nu este necesar să vă încărcați scriptul; nu trebuie decât să creați scriptul în catalogul adecvat indicat de administratorul dumneavoastră de sistem. Dacă folosiți un server Microsoft Windows situat în aceeași rețea locală ca și stația dumneavoastră de lucru, atunci este posibil ca administratorul de sistem să fi alocat o partiție de fișiere în acest scop. În acest caz, încărcarea scriptului PHP se reduce la copierea fișierului care conține scriptul dumneavoastră în server prin tragerea și plasarea fișierului în partiția de fișiere furnizată.

Dacă serverul Windows, UNIX sau Linux nu se află în rețeaua dumneavoastră locală, probabil că veți folosi un program precum FTP pentru a încărca scriptul. În vederea încărcării scriptului dumneavoastră prin intermediul protocolului FTP, solicitați administratorului de sistem următoarele informații:

- Numele gazdei serverului
- Identificatorul dumneavoastră de utilizator și parola pentru deschiderea sesiunii de lucru prin intermediul protocolului FTP
- Catalogul în care trebuie să se găsească scripturile dumneavoastră PHP
- Localizatorul uniform de resurse (URL) pe care trebuie să-l folosiți pentru a obține acces la scripturile dumneavoastră

Pentru a facilita încărcarea scriptului dumneavoastră, poate că preferați să folosiți un client FTP grafic. Puteți găsi programe client FTP gratuite, în sistem shareware sau contra cost în arhive de programe, precum <http://www.tucows.com>. Clientul WS_FTP este un program client FTP foarte popular, adecvat pentru încărcarea scripturilor dumneavoastră. Dacă folosiți un program client FTP grafic, încărcați fișierul prin respectarea instrucțiunilor distribuite o dată cu programul client.

Dacă preferați utilizarea programelor în linie de comandă, puteți folosi clientul FTP inclus în Microsoft Windows, UNIX sau Linux. Programele client existente în fiecare platformă funcționează în moduri mai mult sau mai puțin asemănătoare, deci aceleași instrucțiuni se aplică pentru majoritatea platformelor. Iată care este modul de încărcare a fișierului dumneavoastră.

1. Dacă folosiți Windows, lansați o fereastră de comandă MS-DOS.
2. Folosiți comanda `cd` pentru a vă deplasa în catalogul care conține scriptul dumneavoastră.
3. Emiteți comanda

`ftp gazda` ← `gazda` este numele gazdei serverului PHP

4. Ca răspuns la solicitarea identificatorului dumneavoastră de utilizator, introduceți identificatorul de utilizator furnizat de administratorul dumneavoastră de sistem.
5. Ca răspuns la solicitarea parolei dumneavoastră, introduceți parola-furnizată de administratorul dumneavoastră de sistem.
6. Folosiți comanda `cd` pentru a vă deplasa în catalogul în care trebuie încărcat scriptul dumneavoastră.
7. Emiteți comanda

`Put script` ← `script` este numele fișierului care conține scriptul

8. Emiteți comanda `quit`.

Iată un exemplu de sesiune FTP care folosește un program client și un server Linux. Dacă folosiți o altă platformă, datele dumneavoastră de ieșire vor avea un alt aspect.

```
cd /home/bmccarty/scripts
$ ftp ftp.osborne.com
Connected to ftp.osborne.com.
220 ftp.osborne.com FTP server (Version wu-2.6.0(1)
Wed Jul 26 15:29:19 PDT 2001) ready.
Name (ftp:bmccarty): bmccarty
331 Password required for bmccarty.
Password: xxxxxxxx
230 User bmccarty logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd public_html
250 CWD command successful.
ftp> put test-script.php
local: test-script.php remote: test-script.php
200 PORT command successful.
150 Opening BINARY mode data connection for test-script.php.
226 Transfer complete.
34 bytes sent in 0.000446 secs (74 Kbytes/sec)
ftp> quit
221-You have transferred 34 bytes in 1 files.
221-Total traffic for this session was 498 bytes in 1 transfers.
221-Thank you for using the FTP service on ftp.osborne.com.
221 Goodbye.
$
```



Sfatul specialistului

Întrebare: Există și alte modalități de încărcare a scripturilor, în afară de partițiile de fișiere Windows și protocolul FTP?

Răspuns: Da, există numeroase alte modalități. Uneori, administratorii sistemelor UNIX și Linux le configurează pe acestea de așa manieră încât să vă permită să încărcați scripturi prin intermediul sistemului de fișiere de rețea (Network File System – NFS). Unii administratori de sistem furnizează pagini Web speciale pe care le puteți folosi pentru încărcarea scripturilor. Dacă un server furnizează serviciul Secure Shell Service (SSH), puteți folosi programul scp pentru a vă încărca scripturile. Aceasta este o metodă deosebit de bună, deoarece vă protejează sesiunea de lucru și datele pe care le transferați împotriva „spionilor” din rețea. Transferurile executate cu ajutorul protocolului FTP, de exemplu, trimit identificatorul de utilizator și parola dumneavoastră sub formă de text „în clar”, nu în formă codificată, ceea ce poate duce la o breșă a securității sesiunii de lucru.

Executarea unui script

După ce v-ați încărcat fișierul care conține scriptul, sunteți pregătit pentru a obține accesul la acesta. Lansați-vă browserul Web preferat și deschideți adresa URL asociată scriptului dumneavoastră. Adresa URL trebuie să fie alcătuită din adresa URL identificată de administratorul dumneavoastră de sistem, urmată de un slash (/), urmată de numele fișierului care conține scriptul dumneavoastră. Dacă adresa URL identificată de administratorul dumneavoastră de sistem se încheie deja cu un caracter slash, nu trebuie să mai inserați încă un asemenea caracter înainte de numele scriptului dumneavoastră.

De exemplu, să presupunem că doriți să obțineți accesul la scriptul dumneavoastră încărcat, denumit `test-script.php`. Dacă administratorul dumneavoastră de sistem a identificat `http://www.osborne.com/~bmccarty` ca adresă URL a catalogului care conține scripturile dumneavoastră PHP, puteți obține accesul la scriptul dumneavoastră prin intermediul adresei URL `http://www.osborne.com/~bmccarty/test-script.php`. Dacă administratorul dumneavoastră de sistem a identificat `http://www.osborne.com/~bmccarty/` ca adresă URL a catalogului care conține scripturile dumneavoastră PHP, puteți obține accesul la scriptul dumneavoastră prin intermediul aceleiași adrese URL ca aceea prezentată anterior.

Dacă ați tastat corect adresa URL a scriptului dumneavoastră, iar scriptul respectiv nu conține erori, veți vedea datele de ieșire ale scriptului dumneavoastră. Felicitări! Ați devenit programator PHP!



Proiectul 1-1: Un prim script PHP

În cadrul acestui proiect, veți crea și veți executa un script PHP simplu. Pentru a finaliza proiectul, trebuie să aveți acces la un server care acceptă PHP și încărcarea fișierelor prin intermediul protocolului FTP.

Scopurile proiectului

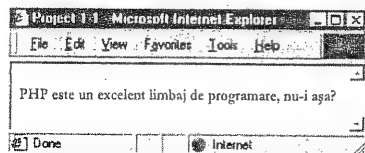
- Crearea unui script PHP
- Încărcarea scriptului PHP într-un server
- Executarea scriptului PHP

Pas cu pas

1. Folosind un editor de texte, creați un script PHP simplu, care trimite date de ieșire sub formă de text la un browser Web. Salvați scriptul într-un fișier denumit p-1-1.proj. Dacă preferați să folosiți inițial un script ambalat, puteți utiliza următorul script:

```
<?php
// PHP: Ghidul incepatorului
// Proiectul 1-1
echo("PHP este un excelent limbaj de programare, nu-i asa?");
?>
```

2. Folosiți protocolul FTP pentru a încărca fișierul care conține scriptul dumneavoastră în catalogul adecvat din serverul dumneavoastră.
3. Dacă este necesar, modificați permisiunile fișierului script astfel încât serverul Web să poată executa scriptul.
4. Folosiți un browser Web pentru a obține accesul la adresa URL asociată fișierului care conține scriptul dumneavoastră. Dacă ați folosit scriptul „ambalat” prezentat în etapa 1, fereastra browserului dumneavoastră Web va avea un aspect oarecum similar celui prezentat în ilustrația următoare.



Depanarea unui script

Uneori, în locul datelor de ieșire ale scriptului dumneavoastră, puteți vedea unul din următoarele:

- Textul scriptului, în loc de datele de ieșire ale acestuia
- O casetă de dialog, prin care sunteți întrebat dacă doriți să descărcați fișierul care conține scriptul
- Un mesaj în care se spune că scriptul nu există
- Un mesaj în care se spune că browserul dumneavoastră Web nu are permisiunea de a obține accesul la script
- Un mesaj în care se spune că scriptul dumneavoastră conține o eroare

La vizualizarea rezultatelor unui script PHP se pot produce numeroase erori, chiar dacă scriptul în sine este corect. Dacă vedeți textul scriptului dumneavoastră

sau o casetă de dialog prin care sunteți întrebat dacă doriți să descărcați fișierul care conține scriptul, este posibil ca extensia fișierului script să fie incorrectă sau ca serverul PHP să nu funcționeze. Deși fișierele script PHP trebuie să aibă, în general, extensia .php, este posibil ca un administrator de sistem să configureze un server PHP astfel încât acesta să impună o altă extensie de fișier. Astfel, dacă scriptul dumneavoastră eșuează din unul dintre aceste două motive, luați legătura cu administratorul dumneavoastră de sistem.

Dacă vedeți un mesaj în care se spune că scriptul nu există, este posibil ca dumneavoastră să fi tastat incorrect adresa URL. Verificați dacă ați tastat corect adresa URL identificată de administratorul dumneavoastră de sistem, precum și dacă ați atașat corect la aceasta numele fișierului care conține scriptul, folosind un slash numai dacă adresa URL identificată de administratorul dumneavoastră de sistem nu se încheie cu un atare caracter.

Dacă vedeți un mesaj în care se arată că browserul dumneavoastră Web nu are permisiunea de a obține accesul la script, poate că este necesar să modificați permisiunile fișierului script. Pentru a afla cum trebuie procedat, consultați-vă cu administratorul de sistem.

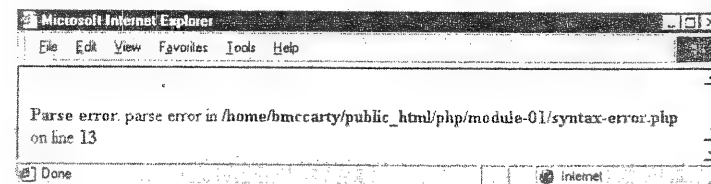
Dacă vedeți un mesaj în care se spune că scriptul dumneavoastră conține o eroare, verificați dacă nu au apărut următoarele probleme:

- O eroare de tastare, cum ar fi scrierea greșită a cuvântului *echo*
- O eroare de punctuație, cum ar fi paranteze, ghilimele duble sau punct și virgulă lipsă sau inserate greșit
- Neincluderea sau includerea eronată a liniilor de delimitare a scriptului, în speță `<?php` și `?>`
- Un marcaj de comentariu (//) care lipsește sau care a fost introdus greșit

De exemplu, iată un script care conține un tip de eroare frecvent întâlnit. Puteți identifica eroarea?

```
<?php
// PHP: Ghidul incepatorului
// Acest script contine o eroare de sintaxa.
echo("Salut, World Wide Web!");
?>
```

Din script lipsește caracterul ghilimele duble de închidere, care trebuie să delimiteze expresia de tip text. Dacă încercați să executați acest script, veți primi o eroare similară celei prezentate în continuare.



Mesajul de eroare încearcă să vă indice sursa erorii, indicând numărul liniei la care s-a produs eroarea. Totuși, remarcați că mesajul vă îndrumă spre linia 13 a unui script care conține numai 5 linii. Din moment ce ghilimelele duble de închidere lipsesc, serverul PHP caută dincolo de sfârșitul scriptului pentru a găsi ghilimelele duble respective. Ca atare, serverul PHP este oarecum derutat cu privire la sursa erorii. Morala este aceea că nu puteți conta în totalitate pe serverul PHP pentru a determina locația erorii; folosiți numărul de linie furnizat de server numai ca îndrumar pentru a depista locația probabilă a erorii. În Modulul 17 veți învăța mai multe noțiuni despre depanarea scripturilor PHP.



Test „la minut”

- Găsiți cele trei erori de sintaxă din următorul script PHP:

```
<?php4
/* PHP: Ghidul incepatorului
/* Acest script contine o eroare de sintaxa.
echo("Salut, World Wide Web!");
?>
```



Test de evaluare

1. Ce program Windows este frecvent folosit pentru crearea scripturilor PHP?
2. Care trebuie să fie prima linie într-un script PHP?
3. Care sunt caracterele ce trebuie folosite pentru a denumi un fișier care conține un script PHP?
4. Care trebuie să fie extensia unui fișier care conține un script PHP?
5. Care este instrucțiunea PHP folosită pentru a trimite date de ieșire sub formă de text unui browser Web?
6. Care este programul frecvent folosit pentru a încărca un script PHP într-un server?

Răspunsuri la test:

- Prima linie trebuie să conțină textul `<?php`, nu `<?php4`.
- Comentariile trebuie să înceapă cu caracterele `//`, nu `/*`.
- Instrucțiunea de reflectare trebuie să se încheie cu un caracter punct și virgulă.

Totuși, deoarece aceasta este ultima linie a scriptului, respectivul va funcționa corect, chiar dacă acel caracter lipsește.

Modulul 2: Elementele constructive ale limbajului PHP

Scopuri

- **Învățați să scrieți numere și șiruri**
- **Învățați să folosiți ghilimele și caractere escape pentru a specifica valori de tip șir speciale**
- **Înțelegeți diferența dintre valori literale și variabile**
- **Învățați modul de utilizare a tablourilor pentru stocarea mai multor valori**
- **Învățați modul de utilizare a operatorilor pentru combinarea valorilor în expresii**
- **Învățați modul de utilizare a funcțiilor pentru executarea operațiilor elementare**

În cadrul acestui modul, veți învăța modul de creare a componentelor care alcătuiesc programele PHP. În modulele 3 și 4 veți învăța modalitățile de asamblare a acestor componente în programe complete.

Numere și șiruri

Programele de calculator manipulează datele, care reprezintă informații. Programele PHP folosesc două categorii principale de date: numere și șiruri. Numerele sunt compuse mai ales din cifre, în timp ce un șir poate conține orice caracter, inclusiv cifre, litere și simboluri speciale.

Decizia privind modul de stocare a datelor este importantă. În mod caracteristic, datele se stochează sub formă de numere atunci când se dorește executarea unor operații matematice asupra datelor, deoarece numerele sunt stocate într-un mod care permite efectuarea de calcule. Pe de altă parte, șirurile sunt stocate folosind o modalitate care facilitează înțelegerea lor de către operatorul uman. Datele trebuie stocate sub formă de șiruri dacă formatul acestora nu este numeric sau dacă doriți ca operatorul uman să fie capabil de a introduce sau de a vizualiza datele. Practic, puteți asimila numerele cu un mod de stocare a datelor în interiorul calculatorului, în speță un *format intern*. Șirurile se pot asimila unui mod de stocare a datelor în afara calculatorului, în speță un *format extern*.

Numere

PHP folosește două categorii de numere: *întregi* și *duble*. Numerele întregi reprezintă numerele fără parte fracționară folosite la numărare, plus zero și numerele negative. Cu alte cuvinte, în PHP termenul de *întreg* are aceeași semnificație ca și în matematică. De exemplu, numărul 100 poate fi reprezentat în PHP sub formă de întreg. Numerele duble, spre deosebire de întregi, reprezintă valori numerice care pot include fracții zecimale, ca de exemplu 2,5. Numerele duble sunt sinonime cu numerele reale din matematică. Uneori, numerele duble mai sunt denumite și *numere cu virgulă mobilă* (în lb. engleză se folosește *punctul zecimal* în loc de virgulă – N. T.).

Deoarece PHP stochează numerele în calculatoare, care dispun de o cantitate limitată de memorie, numerele întregi și duble din PHP diferă de omoloagele lor matematice prin aceea că precizia lor este limitată. În general, numerele întregi sunt stocate sub formă de valori pe 32 de biți, ceea ce le limitează la domeniul cuprins între -2.147.483.648 și 2.147.483.647 inclusiv. Totuși, unele calculatoare stochează numerele PHP întregi într-un mod mai compact, limitând și mai mult domeniul de valori posibile.

În general, numerele duble sunt stocate folosindu-se formatul standard IEEE-64, care furnizează 64 de biți. Acest format vă permite să stocați valori care pot merge până la $1,8 \times 10^{308}$ sub formă de numere duble și furnizează aproximativ 14 cifre după punctul zecimal (sau cifre semnificative) de precizie.

Scrierea numerelor PHP este simplă. Un întreg PHP se obține prin scrierea cifrelor care îi alcătuiesc valoarea. Dacă valoarea este negativă, scrieți un semn minus imediat la stânga numărului. Evitați să scrieți spații sau virgule ca parte a unui întreg PHP. Iată câteva exemple de numere PHP întregi corecte și incorecte:

-3	Corect
0	Corect
5	Corect
2.5	Încorct; conține o parte fracționară
2.0	Încorct; conține o parte fracționară, chiar dacă valoarea acesteia este zero
1,024	Încorct; conține o virgulă
- 7	Încorct; conține un spațiu între semnul minus și cifra
2147483648	Încorct; prea mare

Un număr PHP dublu se scrie cu ajutorul unei serii de cifre, plasând un punct zecimal la locația adecvată. Ca în cazul întregilor PHP, dacă valoarea este negativă, scrieți un semn minus imediat la stânga numărului. De asemenea, din nou similar cu numerele PHP întregi, trebuie să evitați a scrie spații sau virgule ca parte a unui număr dublu. Iată câteva exemple de numere duble corecte și incorecte:

-2.5	Corect
0.0	Corect
3.125	Corect
2	Încorct; îi lipsește punctul zecimal

Când scrieți numere duble foarte mari sau foarte mici, puteți folosi o formă specială, care arată astfel: 2.3e4. Numărul plasat după litera *e* determină înmulțirea cu 10 la puterea dată de numărul respectiv a numărului plasat anterior literei respective. De exemplu, numărul dublu dat anterior are valoarea $2,3 \times 10^4$, iar valoarea dublă $-1.1e-10$ are valoarea $-1,1 \times 10^{-10}$.

Șiruri

Spre deosebire de întregi și de numere duble, care conțin cu precădere cifre, șirurile pot conține orice caracter. Ca atare, șirurile sunt utile pentru stocarea datelor care nu pot fi calculate, precum nume și adrese.

De asemenea, șirurile pot fi utilizate pentru stocarea datelor numerice. Reprezentările sub formă de numere întregi și duble sunt folosite, în general, numai în interiorul calculatoarelor; de regulă, datele sunt introduse în calculatoare și afișate de către acestea sub formă de șiruri. De exemplu, să presupunem că un program convertește temperatura din grade Fahrenheit în grade centigrade. Utilizatorul programului introduce temperatura sub formă de valoare de tip șir. Programul convertește valoarea șir într-o valoare dublă, execută un calcul și convertește rezultatul într-un șir, care este afișat ca rezultat. Motivul derulării acestui proces aparent complicat este acela că sistemele de calcul execută eficient operații aritmetice cu valori întregi și duble; conversia datelor din format șir în format numeric și viceversa este mai simplă decât executarea de operații aritmetice cu șiruri.

Pentru a specifica un șir în PHP, caracterele care alcătuiesc șirul sunt incluse între ghilimele duble; de exemplu, șirul reprezentând numele fizicianului care a formulat teoria relativității este "Albert Einstein". Așa cum s-a explicat, un șir poate conține date numerice; de exemplu, "3.14159". În Modulul 5, veți învăța să converțiți șiruri care conțin date numerice în valori întregi și duble.

PHP facilitează includerea în șiruri a unor caractere speciale, precum caracterele de salt la linie nouă sau retur de car, prin furnizarea de *secvențe escape* care reprezintă caractere speciale. Iată secvențele escape folosite în PHP:

\n	salt la linie nouă
\r	retur de car
\t	caracter de tabulare pe orizontală
\\	backslash
\\$	simbolul dolarului
\"	ghilimele duble

Ca exemplu, iată un șir care include un retur de car, urmat de un salt la linie nouă: "Salut, lume!\r\n". Rețineți că fiecare secvență escape începe cu un backslash (\). Pentru a include un backslash într-un șir, trebuie să folosiți secvența escape adecvată, care este alcătuită din două caractere backslash.



Test „la minut”

- Care este reprezentarea de date cea mai adecvată pentru stocarea consumului mediu de combustibil al unui autoturism, exprimat în mile* pe galon** sau litri pe kilometru?
- Care este reprezentarea de date cea mai adecvată pentru stocarea numărului de capitole al unei cărți?
- Care este reprezentarea de date cea mai adecvată pentru stocarea unui număr de telefon?



Sfatul specialistului

Întrebare: Dacă se dorește includerea unei ghilimele duble în cadrul unui șir? Se poate proceda astfel fără a se folosi o secvență escape?

Răspuns: Simpla inserție a unei ghilimele duble în cadrul unui șir ar deruta limbajul PHP, deoarece ghilimelele duble vor marca, în aparență, sfârșitul șirului. Pentru a include ghilimelele duble în cadrul unui șir, includeți șirul între ghilimele simple, nu duble, astfel: 'El zise "salut."'. Șirurile încadrate între ghilimele simple, respectiv duble, se comportă oarecum diferit unele în raport cu celelalte; veți învăța mai multe despre șiruri în Modulul 9.

Valori literale și variabile

Categoriile de valori despre care ați învățat până acum se numesc *valori literale*. Deseori, este convenabil să atribui un nume unei valori, similar procedurii comun folosit în algebră. O valoare cu nume se numește *variabilă*, deoarece este posibilă modificarea valorii asociate numelui. Prin contrast, o valoare literală este fixă.

Dacă preferați, o variabilă poate fi asimilată cu o casetă care poartă numele variabilei. Valoarea unei variabile este dată de o valoare literală, scrisă pe o bucată de hârtie plasată în interiorul casetei. În orice moment, puteți înlocui bucata de hârtie cu o altă, care conține o nouă valoare a variabilei.

* Unitate de măsură pentru distanțe folosită în țările de limbă engleză și egală cu aproximativ 1,7 km. – N.T.

** Unitate de măsură pentru capacități folosită în țările de limbă engleză și egală cu aproximativ 4 litri. – N.T.

Răspunsuri la test:

- Dublu, deoarece valoarea include deseori o parte fracționară
- Întreg, deoarece valoarea este un număr fără parte fracționară
- Șir, deoarece valoarea nu va fi calculată și poate conține liniuțe, spații sau paranteze

PHP impune câteva reguli asupra numelor variabilelor, astfel încât să poată face imediat diferența dintre variabile și numere, șiruri și alte elemente de program. Iată o metodă de formare a unui nume corect de variabilă PHP:

1. Începeți cu simbolul dolarului (\$).
2. După simbolul dolarului, scrieți o literă sau o liniuță de subliniere (_). Litera poate fi scrisă cu majuscule sau minuscule.
3. Continuați prin a adăuga oricâte litere, cifre sau liniuțe de subliniere doriți. Nu vă lăsați dus de val și să creați un nume de variabilă atât de lung, astfel încât să fie dificil de tastat. Creați, totuși, un nume care să descrie cu claritate scopul variabilei.

Iată câteva exemple de nume de variabile corecte și incorecte:

\$lungime	Corect
\$x	Corect, dar nu foarte descriptiv
y	Încorct, nu începe cu semnul dolarului
\$1side	Încorct, semnul dolarului nu este urmat de o literă sau de un caracter de subliniere
\$a+b	Încorct, conține semnul plus acolo unde sunt permise numai litere, cifre și caractere de subliniere

Deși puteți folosi litere majuscule sau minuscule în numele variabilelor, diferența dintre literele scrise cu majuscule și cele scrise cu minuscule este importantă.

Variabila denumită \$A nu este una și aceeași cu variabila \$a.

Pentru a asocia o valoare unei variabile, veți scrie ceea ce se numește o *instrucțiune de atribuire*. Iată un exemplu simplu:

```
$temperatura = 72.3;
```

Numele variabilei este urmat de un semn egal (=), care identifică instrucțiunea ca fiind o instrucțiune de atribuire. Semnul egal este urmat de valoarea care urmează a fi atribuită variabilei. În acest exemplu, valoarea este dată de valoarea literală dublă 72.3. Caracterul punct și virgulă (;) marchează sfârșitul instrucțiunii.

Exemplul anterior a atribuit unei variabile o valoare literală. De asemenea, puteți atribui valoarea unei variabile către o altă variabilă, prin scrierea unei instrucțiuni de atribuire astfel:

```
$scastigator = $punctajul_cel_mai_mare;
```

În acest caz, valoarea variabilei \$punctajul_cel_mai_mare înlocuiește valoarea variabilei \$scastigator. Ulterior pe parcursul acestui modul, veți învăța să scrieți instrucțiuni de atribuire mai sofisticate.

Ca o valoare literală, o variabilă poate avea o valoare de tip întreg, dublu sau șir. Forma valorii unei variabile se numește *tipul variabilei*. Tipul unei variabile se poate modifica dacă atribuiți variabilei o valoare de un tip diferit față de cel al valorii curente a variabilei. De exemplu, instrucțiunea de atribuire

```
$x = 1;
```

atribuie variabilei \$x tipul întreg. Dacă instrucțiunea de atribuire

```
$x = 1.5;
```

va fi executată ulterior, variabila \$x devine de tip dublu. În multe limbaje de programare, tipul unei variabile nu poate fi modificat. Dacă ați programat folosind un asemenea limbaj, la început s-ar putea ca această caracteristică a limbajului PHP să vi se pară deconcertantă, dar probabil că o veți găsi extrem de convenabilă după ce vă veți fi acomodat cu ea.



Test „la minut”

- Atribuiți un nume adecvat unei variabile care stochează distanța până la Soare.
- Scrieți o instrucțiune de atribuire care atribuie valoarea 3.14159 unei variabile denumite \$pi.



Sfatul specialistului

Întrebare: În ce mod diferă instrucțiunile de atribuire PHP de ecuațiile matematice?

Răspuns: Deși instrucțiunile de atribuire din limbajul PHP și ecuațiile matematice folosesc ambele semnul egal, cele două noțiuni sunt foarte diferite, deoarece atribuirea nu este același lucru cu egalitatea. Atribuirea este o *operație* care înlocuiește o valoare cu o altă. Pe de altă parte, egalitatea este o *relație* între două valori. Când două valori sunt egale, acestea rămân egale pentru totdeauna. Totuși, puteți atribui o valoare unei variabile și ulterior puteți atribui aceleiași variabile o altă valoare. Cu alte cuvinte, egalitatea este permanentă; atribuirea nu este.



Proiectul 2-1: Vizualizarea valorilor variabilelor PHP

În cadrul acestui proiect, veți crea și veți executa un script PHP simplu care demonstrează modul de utilizare a valorilor literale, a variabilelor și a instrucțiunilor de atribuire.

Scopurile proiectului

- Crearea unui script PHP care conține mai multe instrucțiuni de atribuire și instrucțiuni echo
- Încărcarea și executarea scriptului

Răspunsuri la test:

- \$distanța_la_soare sau ceva similar
- \$pi = 3.14159;

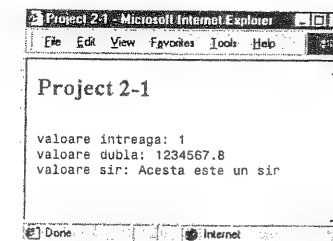
Pas cu pas

1. Folosind un editor de texte, creați un fișier care conține următorul script PHP:

```
<?php
// PHP: Ghidul incepatorului
// Proiectul 2-1

$valoare_intreaga = 1;
$valoare_dubla = 1.2345678e6;
$valoare_sir = "Acesta este un sir";
echo("<H2>Proiectul 2-1</H2>");
echo("<BR>valoare intreaga: ");
echo($valoare_intreaga);
echo("<BR>valoare dubla: ");
echo($valoare_dubla);
echo("<BR>valoare sir: ");
echo($valoare_sir);
?>
```

2. Încărcați fișierul care conține scriptul dumneavoastră în catalogul adecvat al serverului.
3. Folosiți un browser Web pentru a obține accesul la adresa URL asociată fișierului care conține scriptul dumneavoastră. Fereastra browserului dumneavoastră Web trebuie să prezinte un aspect similar celui din ilustrația următoare.



Valori scalare și tablouri

Majoritatea cumpărătorilor preferă să cumpere ouăle în ambalaje de câte 10, nu unul câte unul. Similar, deseori este convenabilă stocarea mai multor valori într-o variabilă. O asemenea variabilă se numește *tablou*, iar valorile individuale se numesc *elementele* tabloului. Variabilele care au o singură valoare se numesc *scalare*. Pentru a fi posibil accesul individual la fiecare element al unui tablou, fiecare element are o *cheie* asociată. Dacă preferați, puteți asimila numele unui tablou cu numele de familie al tuturor elementelor sale. Similar, cheia unui element este echivalentă cu numele de botez al elementului respectiv. Combinația între numele tabloului (numele de familie) și valoarea unei chei (numele de botez) identifică un element al tabloului.

Pentru a crea un tablou, atribuiți unui element al tabloului o valoare și o cheie. De exemplu, instrucțiunea de atribuire

```
$clasa[1] = "geometrie";
```

crează un tablou denumit \$clasa și un element cu valoarea "geometrie" identificat prin cheia 1. Pentru a stoca în tablou o a doua valoare, puteți folosi următoarea instrucțiune de atribuire:

```
$clasa[2] = "contabilitate";
```

Pentru a obține acces la un element al tabloului, specificați numele tabloului și valoarea cheii. De exemplu, instrucțiunea de atribuire

```
$clasa_mare = $clasa[1];
```

atribuie valoarea "geometrie" variabilei scalare \$clasa_mare.

Cheile folosite pentru identificarea elementelor unui tablou nu trebuie să fie numere consecutive; nici măcar nu trebuie să fie numere. De exemplu, iată instrucțiuni de atribuire care creează un tablou ce stochează preferințe în materie de înghețată:

```
$preferinta["Nelu"] = "inghetata elvetiana cu migdale simpla";
$preferinta["Gina"] = "capsuni";
```

Tabloul înregistrează faptul că Nelu preferă înghețata elvețiană simplă cu migdale, iar Gina preferă înghețată cu căpșuni. Un asemenea tablou simplifică determinarea preferințelor în materie de înghețată ale unei persoane, dat fiind prenumele acesteia. Elementele unui tablou cu chei non-numerice sunt accesibile în același mod ca și elementele unui tablou cu chei numerice. De exemplu, instrucțiunea de atribuire

```
$specialitatea_zilei = $preferinta["Nelu"];
```

atribuie variabilei \$specialitatea_zilei valoarea "inghetata elvetiana cu migdale simpla".



Test „la minut”

- Scrieți instrucțiuni de atribuire care creează un tablou denumit \$dimensiune, în cadrul căruia valorile *mic*, *mediu* și *mare* sunt asociate cheilor 1, 2, respectiv 3.
- Scrieți instrucțiuni de atribuire care creează un tablou denumit \$nume_judet, care vă permite să determinați numele complet al unui județ din România în funcție de abrevierea numelui județului folosită pe plăcuțele de înmatriculare ale autoturismelor. Pentru a evita complicațiile, puteți include numai județele Prahova, Dolj și Teleorman.

Răspunsuri la test:

```
$dimensiune[1] = "mic";
$dimensiune[2] = "mediu";
$dimensiune[3] = "mare";

$nume_judet["PH"] = "Prahova";
$nume_judet["DJ"] = "Dolj";
$nume_judet["TR"] = "Teleorman";
```

Evident, textul testului a fost adaptat. În original se face referire la unele state din componența S.U.A. – N.T.

Operatori și funcții

Pentru a vă ajuta să efectuați calcule și prelucrări ale datelor, PHP include o diversitate de operatori și funcții utile. Când combinați valorile literale și variabilele cu operatori și funcții, construiți ceea ce este cunoscut sub numele de *expresii*.

Operatori

PHP include operatorii familiari folosiți pentru executarea operațiilor aritmetice:

+	Adunare
-	Scădere
*	Înmulțire
/	Împărțire

Utilizarea asteriscului (*) ca simbol al înmulțirii poate părea neobișnuită; totuși, este un simbol frecvent folosit în limbajele de programare, deoarece previne confuzia care poate apărea dacă în locul acestuia ar fi fost folosită litera *x*.

Iată câteva exemple simple care demonstrează utilizarea operatorilor pentru a efectua calcule și pentru a atribui valori variabilelor:

```
$profit = $vanzari - $cheltuieli;
$arie = $inaltime * $latime;
$circumferinta = 3.14159 * $diametru;
$rata_impozit = $impozit / $venit_impozabil;
```

Variabilele sau valorile literale asociate cu un operator se numesc *operanzi*. De exemplu, operanzii operatorului de scădere din prima instrucțiune prezentată ca exemplu sunt variabilele \$vanzari și \$cheltuieli.

O proprietate interesantă a operatorului de împărțire este aceea că returnează o valoare întreagă dacă ambii săi operanzi sunt întregi, iar rezultatul este un întreg; în caz contrar, returnează o valoare cu virgulă mobilă. Astfel, instrucțiunea de atribuire

```
$x = 10 / 3;
```

atribuie valorii \$x valoarea cu virgulă mobilă 3.3333333333333, chiar dacă operanzii operatorului de împărțire sunt ambii întregi.

În afară de acești operatori aritmetici familiari, PHP include numeroși operatori mai puțin cunoscuți:

%	Modulo
++	Incrementare
--	Decrementare
.	Concatenare

Similar operatorului de împărțire, *operatorul modulo* execută o împărțire; cu toate acestea, operatorul modulo returnează restul, nu câtul împărțirii. De exemplu, prin împărțirea lui 10 la 3 se obține câtul 3 și restul 1. Deci, instrucțiunea de atribuire


```
$x = 10 % 3;
```

atribuie variabilei \$x valoarea 1.

În programare, operațiile de adăugare, respectiv de scădere a unei unități dintr-o valoare sunt frecvent întâlnite. Pentru comoditate, PHP include operatori care execută aceste operații. *Operatorul de incrementare* adaugă o unitate la valoarea unei variabile, iar *operatorul de decrementare* scade o unitate din valoarea unei variabile. Operatorii sunt utilizați astfel:

```
++$x;
--$y;
```

Rețineți că acești operatori au nevoie de un singur operand, în timp ce majoritatea operatorilor necesită doi operanzi. Prima instrucțiune adaugă o unitate la valoarea variabilei \$x, în timp ce a doua instrucțiune scade o unitate din valoarea variabilei \$y. Dacă preferați, puteți folosi acești operatori în instrucțiuni de atribuire, după cum urmează:

```
$x = ++$y;
```

Această instrucțiune adaugă o unitate la valoarea variabilei \$y și atribuie valoarea rezultantă variabilei \$x.

? Sfatul specialistului

Întrebare: Ce se întâmplă dacă se plasează un operator de incrementare sau de decrementare după operandul aferent acestuia?

Răspuns: Dacă se plasează un operator de incrementare sau de decrementare după operandul aferent, și nu înaintea acestuia, efectul este ușor diferit. Să examinăm următorul exemplu:

```
$x = $y--;
```

Această instrucțiune scade o unitate din valoarea variabilei \$y, dar atribuie variabilei \$x valoarea originală a variabilei \$y, din care nu s-a scăzut nimic. Prin plasarea unui operator de incrementare, respectiv de decrementare, înaintea unei variabile, se execută o operație de *pre-incrementare*, respectiv *pre-decrementare*, prin plasarea unui operator de incrementare, respectiv de decrementare, după o variabilă, se execută o operație de *post-incrementare*, respectiv *post-decrementare*.

În afară de acești operatori numerici, PHP include un operator de concatenare a șirurilor, denumit uneori *operator cat* sau *operator de unire*, deoarece funcția sa constă în unirea șirurilor. De exemplu, să considerăm următoarele instrucțiuni de atribuire:

```
$nume_botez = "Radu";
$nume_familie = "Vasilescu";
$nume = $nume_botez . " " . $nume_familie;
```

Primele două instrucțiuni de atribuire alocă valori șir unor variabile scalare. Ultima instrucțiune de atribuire folosește operatorul de unire pentru a uni numele de botez cu numele de familie și pentru a insera un spațiu între acestea. Valoarea atribuită variabilei \$nume este "Radu Vasilescu".

Ca în matematică, PHP evaluează operatorii de înmulțire și de împărțire anterior operatorilor de adunare, respectiv scădere. Această caracteristică se numește *precedență*. Datorită precedenței, instrucțiunea

```
$x = 1 + 2 * 3;
```

atribuie variabilei \$x valoarea 7, chiar dacă operatorul de adunare apare înaintea celui de înmulțire. Dacă doriți să controlați precedența unei expresii, puteți folosi paranteze. De exemplu, instrucțiunea

```
$x = (1 + 2) * 3;
```

atribuie variabilei \$x valoarea 9, deoarece partea inclusă între paranteze a expresiei este evaluată prima, așa cum se procedează în algebră.

Funcții

În afară de operatori, PHP include funcții care execută operații utile. Iată unele exemple de funcții:

abs(x)	Returnează valoarea absolută a lui x
ceil(x)	Returnează valoarea x, rotunjită la întregul imediat superior
floor(x)	Returnează valoarea x, rotunjită la întregul imediat inferior
max(x,y,...)	Returnează valoarea maximă a unui set de valori
min(x,y,...)	Returnează valoarea minimă a unui set de valori
pow(x,n)	Returnează numărul x, ridicat la puterea specificată n
strftime(f)	Returnează data curentă, formatată conform conținutului parametrului f
sqrt(x)	Returnează rădăcina pătrată a lui x

În afară de acestea, PHP include multe alte funcții. În Anexa I veți descoperi definiții ale acestor funcții, precum și ale altor funcții PHP frecvent folosite.

Majoritatea funcțiilor necesită una sau mai multe valori de intrare, cunoscute sub numele de *argumente*. De exemplu, funcția sqrt necesită un argument care specifică valoarea a cărei rădăcină pătrată trebuie calculată.

? Sfatul specialistului

Întrebare: Mai include PHP și alți operatori?

Răspuns: Da, PHP include mulți alți operatori în afara celor specificați. În particular, include operatori logici și operatori de comparație, despre care veți învăța în Modulul 6.

Unele funcții, precum `min` și `max`, preiau un număr nedefinit de argumente. Alte funcții nu necesită nici un fel de argumente. Pentru a putea folosi o funcție în mod corespunzător, trebuie să cunoașteți:

- Numele funcției
- Acțiunea funcției și valoarea returnată de aceasta, dacă există
- Numărul argumentelor preluate de funcție
- Semnificația fiecărui argument

Iată un exemplu simplu care folosește o funcție pentru calculul lungimii laturilor unui pătrat, dacă este cunoscută aria pătratului:

```
$lungime = sqrt($arie);
```

Rețineți modul în care argumentul funcției este inclus între paranteze, precum și modul în care funcția și argumentul său sunt folosite într-un mod asemănător cu o valoare literală sau o variabilă. Iată un exemplu care prezintă modul de utilizare a funcției `max`, care preia mai multe argumente:

```
$punctaj_castigator = max($punctaj1, $punctaj2, $punctaj3);
```

Observați că fiecare argument este separat de vecinul său printr-o virgulă.



Test „la minut”

- Scrieți o instrucțiune PHP care adaugă valoarea variabilei `$plata_normala` la aceea a variabilei `$plata_ore_suplimentare` și plasează rezultatul în variabila `$plata_totala`.
- Scrieți o instrucțiune PHP care stochează pătratul valorii variabilei `$x` în variabila `$y`.



Proiectul 2-2: Un calcul în PHP

În cadrul acestui proiect, veți crea și executa un script PHP simplu, care calculează aria unui cerc de rază dată.

Scopurile proiectului

- Crearea unui script PHP care calculează aria unui cerc
- Încărcarea și executarea unui script PHP

Răspunsuri la test:

- `$plata_totala = $plata_normala + $plata_ore_suplimentare;`
- `$y = $x * $x;` sau `$y = pow($x, 2);`

Pas cu pas

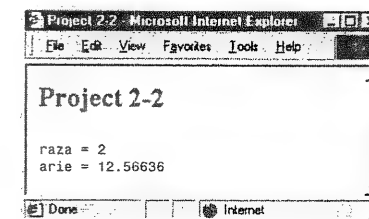
1. Folosind un editor de texte, creați un fișier care conține următorul script PHP:

```
<?php
// PHP: Ghidul incepatorului
// Proiect 2-2
// Calculul ariei unui cerc de raza data

echo("<H2>Proiect 2-2</H2>");
$raza = 2.0;
$pi = 3.14159;
$arie = $pi * $raza * $raza;

echo("raza = ");
echo($raza);
echo("<BR>arie = ");
echo($arie);
?>
```

2. Încărcați fișierul care conține scriptul dumneavoastră în catalogul adecvat al serverului.
3. Folosiți un browser Web pentru a obține accesul la adresa URL asociată fișierului care conține scriptul dumneavoastră. Fereastra browserului dumneavoastră Web ar trebui să aibă un aspect asemănător celui prezentat în ilustrația următoare.



Test de evaluare

1. Scrieți o valoare PHP literală egală cu 12000.
2. Scrieți o valoare PHP literală egală cu 10^{39} .
3. Scrieți o valoare PHP literală care conține numele mărcii autoturismului preferat.
4. Scrieți numele unei variabile PHP adecvate pentru stocarea ratei impozitului aferent vânzărilor curente.
5. Scrieți instrucțiuni PHP care creează un tablou ce asociază numele de botez al fiecăruia dintre membrii familiei dumneavoastră cu anul în care s-a născut persoana respectivă.
6. Scrieți o instrucțiune PHP care calculează circumferința unui cerc pornind de la raza sa, dacă este cunoscută ecuația matematică $C = 2\pi r$ și valoarea aproximativă a lui π egală cu 3,14159.
7. Scrieți o instrucțiune PHP care calculează valoarea absolută a variabilei `$distanța` și stochează rezultatul în variabila `$distanța_neta`.

Modulul 3:

Crearea formularelor HTML

Scopuri

- **Învățați să proiectați formulare HTML eficiente și ușor de utilizat**
- **Învățați să creați formulare HTML care includ controale ce acceptă text și opțiuni ale utilizatorului**
- **Învățați să scrieți programe PHP care obțin accesul la datele ce provin din formularele HTML**

Majoritatea programelor PHP folosesc formulare HTML pentru a prelua datele de la utilizator și pentru a afișa rezultatele calculelor. În cadrul acestui modul, veți învăța să creați formulare HTML care permit programelor dumneavoastră HTML să interacționeze cu utilizatorul.

Noțiuni fundamentale de proiectare a formularelor

Unii programatori experimentați cred că pot determina instantaneu și în mod intuitiv aspectul și modul de funcționare ale oricărui formular HTML. În loc de a proiecta un formular HTML, acești programatori pur și simplu îl creează. Deseori, rezultatele descalifică pretențiile acestor programatori cu privire la cunoștințele obținute intuitiv. Multe formulare de proastă calitate au fost create în acest mod.

O metodă mai bună, atât pentru programatorii neexperimentați, cât și pentru cei cu experiență, este proiectarea formularului anterior creării acestuia. În faza de proiectare, luați în considerare metode alternative de aranjare a formularului și, în cele din urmă, identificați o structură mai bună decât celelalte. Dacă nu ați avut în vedere alternative, nu aveți nici un motiv de a crede că structura formularului dumneavoastră este mai bună decât alte structuri posibile. Ca atare, dacă „săriți” peste faza de proiectare veți obține formulare de calitate inferioară.

Proiectarea unui formular

O modalitate oportună de proiectare a unui formular constă în utilizarea hârtiei și a creionului. O tablă de culoare albă sau neagră este o soluție și mai bună, dacă dispuneți de așa ceva. În cazul în care preferați instrumentele automatizate, puteți folosi un editor HTML de tip WYSIWYG. Toate aceste metode de proiectare au un element comun: se pot șterge cu ușurință o parte sau chiar toate elementele de

proiectare existente și se poate relua activitatea de la început. Cercetări efectuate asupra activității de proiectare au arătat că proiectanții cu experiență o iau de la început mai frecvent decât cei neexperimentați; aceștia din urmă tind să se cantoneze într-o structură inițială și nu iau în considerare alternative potențial mai bune.

Principalele sarcini în proiectarea unui formular HTML le constituie alegerea controalelor HTML care vor fi incluse în formular, selectarea amplasamentului controalelor și alegerea etichetelor acestora. Cu toate acestea, proiectarea nu este un proces de tip „pas cu pas”, care poate fi parcurs în mod mecanic. Este un proces bazat pe oportunitate, ceea ce înseamnă că modalitatea optimă de a parcurge următoarea etapă de proiectare este „în orice mod posibil”. Iată câteva instrucțiuni elementare pentru proiectarea formularelor HTML:

- Alegeți controale care sunt adecvate pentru categoria de date pe care le adună și pentru interacțiunile pe care le furnizează. Secțiunea următoare, care explică fiecare control HTML în parte, vă va ajuta să procedați astfel.
- Etichetați cu claritate fiecare control, astfel încât utilizatorul să înțeleagă funcția acestuia.
- În măsura posibilităților, aliniați etichetele și marginile din stânga ale controalelor. Structura unui formular bine proiectat este asemănătoare cu aceea a unei pagini de ziar, fiind compusă dintr-o serie de coloane în cadrul cărora fiecare element vizual este aliniat cu elementele plasate deasupra, respectiv dedesubtul său. Puteți obține acest tip de structură folosind tabele HTML sau eticheta `BR`.
- Grupați controalele corelate și separați fiecare asemenea grup de celelalte grupuri prin încorporarea de spații albe în structura dumneavoastră.
- Secvența de controale din formular trebuie să fie asemănătoare cu ordinea în care le va manipula utilizatorul. În caz contrar, utilizatorul va fi obligat să piardă timp navigând de la un control la altul.

Este important să rețineți că acestea sunt doar îndrumări, nu reguli. Nefiind reguli, puteți opta pentru încălcarea lor, din când în când. Acest lucru nu constituie o greșală, atât timp cât sunteți convins că rezultatul încălcării unei recomandări este superior rezultatului respectării acesteia. De exemplu, recomandările pot reprezenta îndrumări contradictorii într-o anumită situație. În acest caz, sunteți obligat să încălcați o recomandare sau alta.

Pentru a determina dacă o variantă de proiectare este superioară alteia, este util să cereți părerea unui utilizator. Dacă nici un utilizator nu vă poate fi de ajutor, obțineți asistența unui alt programator. În cel mai rău caz, străduiți-vă să vă puneți dumneavoastră înșivă în rolul unui utilizator al formularului. Întrebați pe utilizator sau pe înlocuitorul acestuia dacă formularul este clar, ușor de utilizat și eficient. În caz afirmativ, este momentul să treceți mai departe și să creați efectiv formularul HTML.

Crearea unui formular HTML

Dacă procesul de proiectare este bazat pe oportunism, procesul de creare a unui formular HTML pornind de la structura dumneavoastră este mai direct. Mai întâi, creați schița HTML elementară, care va conține controalele din formularul dumneavoastră. Apoi, încorporați controalele în structură; în particular, un formular HTML trebuie să includă un buton de expediere pe care utilizatorul execută clic pentru a trimite serverului datele din formular. Deoarece o singură pagină HTML poate conține mai multe formulare, puteți repeta de mai multe ori etapele intermediare ale acestui proces. Sub-secțiunile următoare detaliază procesul de creare a unui formular HTML.

Crearea structurii HTML

Structura HTML pe care o folosiți pentru a include un formular nu este deloc diferită, de fapt, de cea folosită pentru includerea unei pagini HTML obișnuite. De exemplu, iată o structură caracteristică:

```
<HTML>
<HEAD>
<TITLE>Titlul paginii este inserat aici</TITLE>
</HEAD>
<BODY>
  Continutul paginii sau al formularului este inserat aici
</BODY>
</HTML>
```

În interiorul corpului unei pagini HTML care conține un formular puteți folosi orice etichetă HTML obișnuită. Pentru a descrie formularul în sine, folosiți eticheta `FORM`, care are următoarea formă elementară:

```
<FORM METHOD="metoda" ACTION="url">
```

Atributul `METHOD` al etichetei `FORM` poate lua una din valorile `GET` sau `POST`. Pentru moment, specificați întotdeauna valoarea `POST`. Atributul `ACTION` specifică adresa URL a scriptului PHP care prelucrează datele adunate prin intermediul formularului. Adresa URL poate fi o adresă completă, care include protocolul, numele gazdei și calea de acces, respectiv o adresă parțială, care specifică o locație relativă la locația paginii curente. Între eticheta `FORM` și eticheta `sa /FORM` corespunzătoare, plasați controalele formularului.



Sfatul specialistului

Întrebare: Când creez un formular HTML, când trebuie să specific `GET` în loc de `POST` ca valoare a atributului `METHOD`?

Răspuns: Ca începător, este mai bine să folosiți în mod consecvent metoda `POST`, deoarece alegerea între metodele `GET` și `POST` este destul de complicată.

Ca regulă empirică, mulți programatori folosesc `GET` pentru formulare care execută o căutare sau o interogare, respectiv `POST` pentru formulare care actualizează o bază de date sau un fișier. Două dezavantaje ale metodei `GET` sunt acelea că impune o limită asupra cantității de date care pot fi trimise scriptului de prelucrare și că transferă date prin atașarea șirurilor codificate la adresa URL a scriptului de prelucrare. Astfel, datele trimise prin metoda `GET` pot fi vizualizate de către utilizator. Un avantaj al metodei `GET` este acela că utilizatorii pot insera semne de carte în rezultatele unei interogări sau ale unei căutări care folosește metoda `GET`, dar nu pot executa aceeași operație în cazul unei interogări sau al unei căutări care folosește metoda `POST`. Desigur, utilizatorii pot insera un semn de carte la pagina care conține un formular ce folosește metoda `POST`; rezultatele sunt cele unde nu se poate insera semnul de carte.

Încorporarea controalelor

Această sub-secțiune descrie două controale elementare pe care le veți folosi frecvent: caseta cu text și butonul de expediere. În secțiunea următoare, veți afla mai multe despre controalele pe care le puteți folosi în construcția formularelor HTML.

O casetă cu text permite utilizatorului să tasteze informații care pot fi apoi trimise unui script PHP. Casetele cu text sunt frecvent folosite pentru a obține date precum numele utilizatorului, adresa sa poștală sau adresa de e-mail. Pentru a crea o casetă cu text, scrieți o etichetă HTML care are următoarea formă elementară:

```
<INPUT TYPE="TEXT" NAME="text">
```

Atributul `NAME` atribuie casetei cu text un nume, astfel încât conținutul său să fie accesibil unui script PHP. Numele pe care îl atribuiți unui control trebuie să fie unic în cadrul formularului și trebuie să respecte regulile pentru denumirea variabilelor PHP, cu excepția faptului că numele *nu* trebuie să înceapă cu simbolul dolarului. Cu alte cuvinte, numele trebuie să înceapă cu o literă și nu trebuie să conțină alte caractere în afara literelor, a cifrelor și a caracterelor de subliniere; în particular, numele nu trebuie să conțină spații. HTML nu are o etichetă `/INPUT`, deci nu încercați să combinați o etichetă `INPUT` cu o asemenea etichetă.

Un buton de expediere permite utilizatorului să trimită serverului conținutul unui formular. Fiecare formular HTML trebuie să aibă un buton de expediere. Pentru a crea un buton de expediere, scrieți o etichetă HTML care are următoarea formă elementară:

```
<INPUT TYPE="SUBMIT" VALUE="text">
```

Atributul `VALUE` specifică textul care trebuie să apară pe suprafața butonului de expediere.

Iată un formular HTML complet, care include casete cu text ce preiau numele și adresa de e-mail a utilizatorului:

```
<HTML>
<HEAD>
<TITLE>Numele si adresa de e-mail ale utilizatorului</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="phpinfo.php">
<H3>Numele si adresa de e-mail ale utilizatorului</H3>
<BR>Nume:
<BR><INPUT TYPE="TEXT" NAME="numele_utilizatorului">
<BR>Adresa de e-mail:
<BR><INPUT TYPE="TEXT" NAME="adresa_email">
<BR>
<BR><INPUT TYPE="SUBMIT" VALUE="Trimite datele">
</FORM>
</BODY>
</HTML>
```

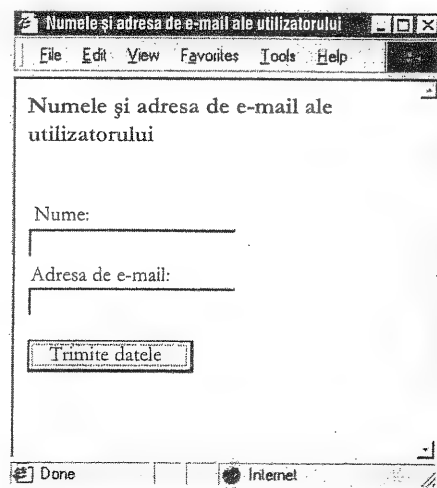
Observați utilizarea etichetelor BR pentru alinierea etichetelor și a controalelor, precum și numele atribuite controalelor de tip casetă cu text. Când utilizatorul execută clic pe butonul de expediere, datele introduse de utilizator sunt trimise scriptului phpinfo.php. Ilustrația alăturată prezintă aspectul formularului.

Lucrul cu formulare multiple

În interiorul corpului unei pagini HTML se pot include mai multe formulare. Dacă procedați astfel, nu uitați să inserați o etichetă </FORM> anterior etichetei <FORM> a următorului formular. Browserul utilizatorului va face indigestie dacă suprapuneți formulare într-o pagină.

Iată un exemplu simplu de pagină care conține mai multe formulare:

```
<HTML>
<HEAD>
<TITLE>Numele si adresa de e-mail ale utilizatorului</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="phpinfo.php">
<H3>Numele si adresa de e-mail ale utilizatorului</H3>
<BR>Nume:
<BR><INPUT TYPE="TEXT" NAME="numele_utilizatorului">
<BR>Adresa de e-mail:
```



```
<BR><INPUT TYPE="TEXT" NAME="adresa_email">
<BR>
<BR><INPUT TYPE="SUBMIT" VALUE="Trimite datele">
</FORM>
<FORM METHOD="POST" ACTION="phpinfo.php">
<H3>Numerele de telefon si fax ale utilizatorului</H3>
<BR>Numar de telefon:
<BR><INPUT TYPE="TEXT" NAME="telefon">
<BR>FAX:
<BR><INPUT TYPE="TEXT" NAME="fax">
<BR>
<BR><INPUT TYPE="SUBMIT" VALUE="Trimite datele">
</FORM>
</BODY>
</HTML>
```

Când utilizatorul execută clic pe butonul de expediere, datele incluse în câmpurile formularului care conține butonul pe care s-a executat clic sunt trimise la server. Astfel, serverul primește fie un nume și o adresă de e-mail, fie numerele de telefon și de fax, nu conținutul tuturor celor patru câmpuri.



Sfatul specialistului

Întrebare: Dacă aptitudinile mele în materie de HTML sunt reduse sau „ruginite”? Unde îmi pot revizui cunoștințele de HTML pentru a putea crea formulare?

Răspuns: Pentru a învăța sau pentru a recapitula elementele fundamentale ale limbajului HTML, vă propun cartea lui Wendy Willard *HTML: Ghidul începătorului* (Osborne, 2000)*. Ca și alte volume din seria *Ghidul începătorului*, cartea lui Wendy conține teste „la minut” și proiecte care contribuie la consolidarea materialului citit. De asemenea, cartea mai conține module referitoare la JavaScript și foile de stil în cascadă, tehnologii care pot fi utilizate eficient alături de PHP.



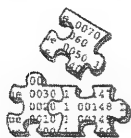
Test „la minut”

- Care sunt atributele care trebuie specificate în eticheta FORM?
- Care este controlul pe care trebuie să-l conțină toate formularele HTML?
- Care este eticheta HTML folosită pentru crearea unei casete cu text?

* Sau oricare dintre excelentele cărți de HTML traduse și publicate de Editura Teora. – N.T.

Răspunsuri la test:

- Atributele METHOD și ACTION
- Butonul de expediere
- INPUT



Proiectul 3-1: Vizualizarea câmpurilor unui formular

În cadrul acestui proiect, veți învăța să vizualizați datele transmise de un formular HTML către scriptul său de prelucrare. Veți descoperi utilitatea acestei caracteristici atunci când trebuie să depanați un formular HTML sau un script de prelucrare.

Scopurile proiectului

- Exersarea creării unui script PHP și a unui formular HTML
- Capacitatea de a sesiza rezultatele expedierii unui formular HTML

Pas cu pas

1. Plasati următorul script HTML într-un fișier denumit p-3-1.php și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
// Fișierul p-3-1.php
phpinfo()
?>
```

2. Plasati următoarea pagină HTML într-un fișier denumit p-3-1.html și încărcați acest fișier în serverul dumneavoastră PHP, plasându-l în același catalog ca și fișierul p-3-1.php:

```
<HTML>
<HEAD>
<TITLE>Numele si adresa de e-mail ale utilizatorului</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-3-1.html -->
<FORM METHOD="POST" ACTION="p-3-1.php">
<H3>Numele si adresa de e-mail ale utilizatorului</H3>
<BR>Nume:
<BR><INPUT TYPE="TEXT" NAME="numele_utilizatorului">
<BR>Adresa de e-mail:
<BR><INPUT TYPE="TEXT" NAME="adresa_email">
<BR>
<BR><INPUT TYPE="SUBMIT" VALUE="Trimite datele">
</FORM>
</BODY>
</HTML>
```

3. Orientați un browser Web spre adresa URL a fișierului care conține formularul HTML. Introduceți numele unui utilizator și adresa sa de e-mail, apoi executați clic pe butonul de expediere.
4. Când scriptul de prelucrare p-3-1.php este executat, acesta afișează un raport amănunțit cu privire la starea serverului PHP. În secțiunea intitulată „Variabile PHP”, puteți găsi informații cu privire la cele două controale de tip casetă cu text, în speță nume_utilizator și adresa_email, așa cum se poate vedea în ilustrația următoare. Remarcați valorile introduse de utilizatorul formularului.

Variable	Value
PHP_SELF	/~bmccarty/php/module-03/project-3-1a.php
HTTP_POST_VARS[user_name]	Bill McCarty
HTTP_POST_VARS[email_address]	mccartyb@osborne.com

Crearea controalelor formularului

În secțiunea anterioară, ați învățat să creați controalele de tip casetă cu text și buton de expediere. În secțiunea de față și în cea următoare, veți extinde repertoriul controalelor dumneavoastră astfel încât să includeți întreaga gamă de controale disponibile.

Crearea casetelor cu text personalizate

Secțiunea precedentă a prezentat sintaxa elementară pentru crearea unei casete cu text. Cu toate acestea, HTML furnizează numeroase atribute suplimentare care vă permit să modificați aspectul și comportamentul unei casete cu text. Iată sintaxa completă pentru crearea unei casete cu text:

```
<INPUT TYPE="TEXT" NAME="text" SIZE="numar" MAXLENGTH="numar" VALUE="text">
```

Atributele TYPE și NAME sunt obligatorii; celelalte atribute sunt opționale. Atributul SIZE, care este dat sub forma unui număr de caractere, stabilește dimensiunea vizibilă a casetei cu text; atributul MAXLENGTH specifică numărul maxim de caractere pe care utilizatorul are permisiunea de a le tasta. Valoarea atributului MAXLENGTH o poate depăși pe aceea a atributului SIZE, în care caz conținutul controlului defilează spre dreapta atunci când utilizatorul a introdus SIZE caractere. Atributul VALUE constituie o valoare care este afișată inițial în caseta cu text.

Sugestie

Este important să cunoașteți aspectul și modul de comportare a casetei cu text și a altor controale HTML de formular. Proiectul care apare la sfârșitul acestei secțiuni vă va oferi posibilitatea de a acumula experiență în crearea controalelor și în lucrul cu acestea.

Crearea de suprafețe cu text

Ca o casetă cu text, o suprafață cu text permite unui utilizator să introducă text. Cu toate acestea, o suprafață cu text poate permite utilizatorului să introducă mai multe linii de text, în timp ce o casetă cu text permite utilizatorului să introducă o singură linie. Iată sintaxa pentru crearea unei suprafețe cu text:

```
<TEXTAREA NAME="text" ROWS="numar" COLS="numar" WRAP="wrap">
```

Atributele NAME și ROWS sunt obligatorii; atributele COLS și WRAP sunt opționale. Atributul ROWS specifică numărul liniilor de text vizibile în suprafața cu text; deoarece suprafața de text defilează după ce a fost umplută, utilizatorul poate introduce linii suplimentare. Atributul COLS specifică numărul coloanelor de text vizibile în suprafața cu text; deoarece suprafața cu text se derulează sau se înfășoară după ce s-a umplut, utilizatorul poate introduce linii mai lungi. Atributul WRAP specifică maniera de înfășurare a textului în interiorul suprafeței cu text. Atributul WRAP poate avea următoarele valori:

Valoare	Descriere
off	Înfășurarea cuvintelor la sfârșitul liniei de text este dezactivată; utilizatorul poate introduce caractere de sfârșit de linie pentru a forța înfășurarea liniilor
virtual	Liniile par înfășurate, dar caracteristica de înfășurare a liniilor nu este trimisă la server
physical	Înfășurarea cuvintelor la sfârșitul liniei de text este activată
soft	Identice cu virtual
hard	Identice cu physical

O etichetă TEXTAREA trebuie combinată cu o etichetă /TEXTAREA. Orice text care apare între etichete va fi prezentat drept conținut inițial al controlului de tip suprafață cu text.

Crearea casetelor cu parolă

Dacă o aplicație impune unui utilizator să introducă o parolă, puteți crea o casetă cu text în acest scop. Totuși, când utilizatorul introduce parola, orice persoană aflată în apropiere poate vizualiza parola, situație care duce la o posibilă breșă de securitate. În loc de a se folosi o casetă cu text pentru introducerea de informații confidențiale, trebuie să folosiți o casetă cu parolă. Pentru a crea o casetă cu parolă, folosiți aceeași sintaxă ca și cea utilizată pentru crearea unei casete cu text, cu excepția faptului că specificați PASSWORD (parolă) în loc de TEXT ca valoare a atributului TYPE:

```
<INPUT TYPE="PASSWORD" NAME="text" SIZE="numar" MAXLENGTH="numar" VALUE="text">
```

Atributele unei casete cu parolă au aceeași semnificație ca și acelea ale unei casete cu text.

Crearea casetelor de validare

Pentru date care pot avea numai una din două valori, cum ar fi „pornit” sau „oprit”, caseta de validare este controlul ideal. De exemplu, caseta de validare este un control adecvat pentru a permite utilizatorului să opteze pentru livrarea rapidă a unui colet. În cazul în care caseta de validare corespunzătoare este validată, coletul va fi livrat mai rapid; în caz contrar, coletul se va deplasa cu mijloacele obișnuite.

Pentru a crea o casetă de validare, folosiți următoarea sintaxă:

```
<INPUT TYPE="CHECKBOX" NAME="text" CHECKED VALUE="text">
```

Atributul TYPE este obligatoriu; atributele NAME, CHECKED și VALUE sunt opționale. Dacă atributul CHECKED apare, caseta de validare va fi selectată în mod prestabilit; în caz contrar, caseta de validare nu este selectată inițial. Atributul VALUE specifică valoarea care este trimisă serverului în cazul în care caseta de validare este selectată; dacă atributul nu este specificat, se va trimite valoarea on (activat).

Crearea butoanelor radio

Ca și casetele de validare, butoanele radio pot avea numai una din două valori. Cu toate acestea, butoanele radio sunt organizate în grupuri, iar la un moment dat poate fi activat un singur buton radio din cadrul unui grup; toate celelalte trebuie să fie dezactivate. Butoanele radio sunt utile pentru a permite unui utilizator să aleagă dintr-o serie de alternative mutual exclusive. De exemplu, puteți folosi un set de trei butoane radio pentru a permite utilizatorului să specifice tipul de ambalaj pentru cadou: fără ambalaj, cu ambalaj simplu sau sofisticat. Numai unul dintre cele trei butoane radio poate fi activat; la un loc, setul de butoane radio oferă utilizatorului o triplă opțiune.

Pentru a crea un buton radio, folosiți următoarea sintaxă:

```
<INPUT TYPE="RADIO" NAME="text" CHECKED VALUE="text">
```

Rețineți că aceasta este aceeași sintaxă folosită pentru crearea unei casete de validare, cu deosebirea că atributul TYPE are valoarea RADIO în locul valorii CHECKBOX. Atributele unui buton radio au aceeași semnificație ca și acelea ale unei casete de validare. Totuși, atributul NAME este obligatoriu pentru butoanele radio, chiar dacă este opțional pentru casetele de validare. Toți membrii unui set de casete de validare prezintă aceeași valoare a atributului NAME.

Crearea de selecții

O *selecție* este un meniu care defilează, de unde utilizatorul poate alege una sau mai multe opțiuni. De exemplu, într-o selecție pot fi enumerate garniturile pentru pizza, astfel încât un utilizator să poată selecta orice combinație de garnituri pe care o dorește. Pentru a crea o selecție, folosiți următoarea sintaxă:


```
<SELECT NAME="text" MULTIPLE SIZE="numar">etichetele OPTION se insereaza aici</SELECT>
```

Eticheta SELECT este folosită în combinație cu eticheta /SELECT. Între cele două etichete este inclusă o serie de etichete OPTION.

Într-o etichetă SELECT, numai atributul NAME este obligatoriu. Atributul MULTIPLE arată că utilizatorul poate alege mai multe opțiuni menținând apăsată tasta CTRL și executând clic pe acestea. În absența atributului MULTIPLE, utilizatorul poate selecta o singură opțiune. Dacă specificați atributul MULTIPLE, trebuie să specificați și un atribut NAME, care atribuie un nume de tablou ca nume al controlului. De exemplu, un control de tip selecție care permite utilizatorului să aleagă mai multe garnituri pentru desert trebuie denumit folosind sintaxa garnitura[], nu garnitura.

Comportarea unei selecții care permite o singură opțiune este echivalentă cu aceea a unui set de butoane radio, dar aspectul unei selecții este diferit de acela al unui set de butoane radio, așa cum veți vedea în proiectul de la sfârșitul acestei secțiuni. Atributul SIZE specifică numărul opțiunilor vizibile. Prin utilizarea unui buton de derulare în jos sau a unei bare de defilare, utilizatorul poate manipula selecția pentru a obține accesul la restul opțiunilor și a alege dintre acestea.

Așa cum s-a arătat, o selecție este asociată cu una sau mai multe opțiuni. Pentru a crea o opțiune care urmează a fi utilizată în cadrul unei selecții, folosiți următoarea sintaxă:

```
<OPTION SELECTED VALUE="text">continutul opțiunii este inserat aici</OPTION>
```

Eticheta OPTION este combinată cu o etichetă /OPTION. Textul dintre aceste etichete este cunoscut sub numele de conținut al opțiunii. Conținutul opțiunii apare în controlul SELECT. Mulți programatori HTML omit eticheta /OPTION, caz în care textul opțiunii se extinde până la următoarea etichetă OPTION sau /SELECT. Totuși, s-ar putea ca acest mod de utilizare să nu fie compatibil cu versiunile ulterioare ale standardului HTML.

Ambele atribute ale etichetei OPTION sunt opționale. Dacă apare atributul VALUE, valoarea sa este trimisă serverului atunci când este selectată opțiunea asociată; în caz contrar, conținutul opțiunii este trimis la server. Atributul SELECTED arată că opțiunea corespunzătoare este selectată inițial; în caz contrar, opțiunea respectivă nu este selectată inițial.

Crearea câmpurilor ascunse

Uneori este utilă crearea așa-numitelor câmpuri ascunse. Valorile câmpurilor ascunse sunt trimise la server alături de valorile altor controale; cu toate acestea, utilizatorul nu are posibilitatea de a vizualiza sau manipula valorile câmpurilor ascunse.

Câmpurile ascunse se utilizează frecvent în cadrul unei serii de formulare. De exemplu, datele introduse de utilizator în primul formular din serie pot fi necesare

în formularele ulterioare. În loc de a determina utilizatorul să introducă datele în fiecare formular, datele pot fi stocate într-un câmp ascuns, creat și inițializat de scriptul care prelucrează primul formular. Astfel, utilizatorul este scutit de o muncă suplimentară și sunt preîntâmpinate eventualele erori, care ar fi putut apărea dacă utilizatorul nu ar fi introdus aceleași date în formularele ulterioare.

Pentru a crea un câmp ascuns, folosiți următoarea sintaxă:

```
<INPUT TYPE="HIDDEN" NAME="text" VALUE="text">
```

Numai atributele TYPE și NAME sunt obligatorii; cu toate acestea, controlul este practic inutil în absența atributului VALUE, a cărui valoare este trimisă în mod automat la server în momentul expedierii formularului.



Sfatul specialistului

Întrebare: Am văzut formulare HTML care permit unui utilizator să expedieze serverului conținutul unui fișier. Cum se poate realiza acest lucru?

Răspuns: Puteți permite utilizatorului să aleagă un fișier și apoi să trimită serverului conținutul fișierului creând un control de tip fișier, prin intermediul următoarei sintaxe:

```
<INPUT TYPE="FILE" NAME="nume" ACCEPT="tip_mime" VALUE="text">
```

Atributul TYPE este singurul obligatoriu; cu toate acestea, în general, apare și atributul NAME. Atributul VALUE specifică un nume de fișier prestabilit. Atributul ACCEPT specifică formatul conținutului fișierului. Pot fi indicate mai multe formate, dar fiecare format trebuie separat de vecinii săi prin intermediul unei virgule. Formatul este specificat folosind un cod MIME (Multipurpose Internet Mail Extensions). Tabelul următor descrie formatele folosite cel mai frecvent:

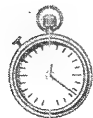
Tip MIME	Tip de date	Extensii de fișier
application/msexcel	Microsoft Excel	xls
application/msword	Microsoft Word	doc, dot
application/octet-stream	Binară	exe
application/pdf	Adobe Acrobat	pdf
application/postscript	Postscript	ai, eps, ps
application/ppt	Microsoft PowerPoint	ppt
application/zip	Date comprimate în format ZIP	zip
audio/midi	Musical Instrument Digital Interface (MIDI)	mid, midi
audio/x-wav	Windows Audio Format (WAV)	wav
image/gif	Compuserve Graphics Interchange Format (GIF)	gif
image/jpeg	Joint Photographics Expert Group (JPEG)	jpeg, jpg, jpe

image/TIFF	Tagged Image Format (TIF)	tif, tiff
text/HTML	HTML	htm, html
text/plain	Text simplu	txt
text/richtext	Rich Text Format (RTF)	rtf
video/mpeg	Secvență video	mpg, mpv, mpe, mpeg
video/quicktime	Secvență video	qt, mov
video/x-msvideo	Secvență video	avi

Eticheta FORM de delimitare trebuie să aibă POST ca valoare a atributului său METHOD. De asemenea, trebuie să includă un atribut ENCTYPE cu valoarea multipart/form-data. Iată sintaxa pe care trebuie să o folosiți:

```
<FORM METHOD="POST" ENCTYPE="multipart/form-data" ACTION="url">
```

Este posibil ca serverul dumneavoastră să nu fie configurat astfel încât să accepte fiecare tip MIME. Consultați-vă cu administratorul serverului dumneavoastră, pentru a determina dacă tipul de date pe care doriți să-l folosiți este acceptat în mod corespunzător.



Test „la minut”

- Scrieți un program HTML care creează o casetă cu text cu lățimea egală cu 40 de caractere și care permite utilizatorului să introducă maximum 80 de caractere.
- Scrieți un program HTML care creează o suprafață cu text având lățimea egală cu 80 de coloane, înălțimea de 10 rânduri și care acceptă înfășurarea textului la sfârșitul liniei de text.
- Scrieți un program HTML care creează o casetă de validare selectată inițial și care trimite serverului valoarea „la modă”.
- Scrieți un program HTML care creează o pereche de butoane radio care vă permit să specificați valoarea „adevărat” sau valoarea „fals”.



Proiect 3-2 O casetă HTML aglomerată

În acest proiect, veți crea o casetă aglomerată, un formular HTML care conține o varietate de controale. Veți putea manipula controalele și vizualiza datele pe care le trimite acestea unui script de prelucrare asociat formularului. Acest proiect este important pentru a dobândi o înțelegere solidă a fiecărui control folosit în formularele HTML, inclusiv aspectul și comportamentul acestuia.

Răspunsuri la test:

- `<INPUT TYPE="TEXT" NAME="xxxx" SIZE="40" MAXLENGTH="80">`
- `<TEXTAREA NAME="xxxx" COLS="80" ROWS="10" WRAP="physical"></TEXTAREA>`
- `<INPUT TYPE="CHECKBOX" NAME="xxxx" CHECKED VALUE="la moda">`
- `<INPUT TYPE="RADIO" NAME="xxxx" VALUE="adevarat">` și `<INPUT TYPE="RADIO" NAME="xxxx" VALUE="fals">`

Scopurile proiectului

- Acumularea de experiență în crearea controalelor care alcătuiesc un formular HTML
- Prezentarea aspectului și comportamentului controalelor care alcătuiesc un formular
- Furnizarea unui mod de a exersa utilizarea atributelor și valorilor asociate etichetelor HTML care precizează controalele dintr-un formular

Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit p-3-2.php și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
// Fișierul p-3-2.php
phpinfo()
?>
```

2. Plasați următoarea pagină HTML într-un fișier denumit p-3-2.html și încărcați acest fișier în serverul dumneavoastră PHP, plasându-l în același catalog ca și fișierul p-3-2.php:

```
<HTML>
<HEAD>
<TITLE>O caseta aglomerata</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-3-2.html -->
<FORM METHOD="POST" ACTION="p-3-2.php">
<H3>O caseta ocupata</H3>
<BR>Casetă cu text:
<BR><INPUT TYPE="TEXT" NAME="casetă_cu_text">
<BR>
<BR>Casetă cu parola:
<BR><INPUT TYPE="PASSWORD" NAME="casetă_cu_parola">
<BR>
<BR>Suprafața cu text:
<BR>
<TEXTAREA NAME="suprafata_cu_text" ROWS="5" COLS="40"
WRAP="physical">
Aici introduceți adresa dumneavoastră.
</TEXTAREA>
<BR>
<BR>Casete de validare:
<BR><INPUT TYPE="CHECKBOX"
NAME="mic_dejun">Mic dejun
<BR><INPUT TYPE="CHECKBOX"
NAME="dejun">Dejun
<BR><INPUT TYPE="CHECKBOX"
NAME="cina">Cina
<BR>
<BR>Butoane radio:
<BR><INPUT TYPE="RADIO" NAME="butonradio"
VALUE="francez" CHECKED>Francez
<BR><INPUT TYPE="RADIO" NAME="butonradio"
VALUE="italian" CHECKED>Italian
<BR><INPUT TYPE="RADIO" NAME="butonradio"
VALUE="rusesec" CHECKED>Rusesec
<BR>
<BR>
```

Selectați:

```
<BR><SELECT MULTIPLE NAME="select[]">
<OPTION SELECTED VALUE="branza">Branza
<OPTION VALUE="Extra">Branza Extra
<OPTION SELECTED VALUE="pepperoni">Pepperoni
<OPTION VALUE="carnati">Carnați
<OPTION VALUE="salam">Salam
<OPTION VALUE="ciuperci">Ciuperci
</SELECT>
<BR>
<BR>
<INPUT TYPE="HIDDEN" NAME="comoara"
VALUE="inestimabila">
<BR>
<BR>
<BR><INPUT TYPE="SUBMIT" VALUE="Trimite datele">
</FORM>
</BODY>
</HTML>
```

- Orientați un browser Web spre adresa URL a fișierului care conține formularul HTML. Imaginea afișată pe ecran de browserul dumneavoastră trebuie să fie asemănătoare celei prezentate în figura 3-1. Examinați controalele din formular, observând aspectul acestora. Exersați utilizarea controalelor, pentru a vedea care sunt tipurile de comportament posibile. În momentul în care sunteți mulțumit de valorile pe care le-ați specificat, executați clic pe butonul de expediere pentru a trimite serverului datele din formular.
- Când scriptul de prelucrare p-3-2.php este executat, afișează un raport amănunțit cu privire la starea serverului PHP. În secțiunea intitulată „Variabile PHP”, puteți găsi informații cu privire la valorile variabilelor folosite în formular, așa cum se poate vedea în figura 3-2.
- Acum, când puteți vedea care este modul de funcționare a programului HTML original, modificați-l pentru a determina efectul diferitelor atribute și valori. De exemplu, eliminați atributul MULTIPLE din selecție și încercați să alegeți mai multe garnituri pentru pizza. Pentru ca modificările aduse programului HTML să fie vizibile în browserul dumneavoastră, nu uitați să actualizați pagina după ce ați modificat-o și ați încărcat-o în server.

Alte noțiuni referitoare la expedierea unui formular

În secțiunile anterioare ale acestui modul, ați învățat modul de creare a formularelor HTML care pot trimite date unui server. În această secțiune, veți învăța mai multe despre procesul de expediere a formularelor. În particular, veți învăța să folosiți o imagine în locul unui buton de expediere, să creați un buton de reinițiere, să creați o pagină care conține mai multe formulare și să transferați informații unui script prin intermediul adresei URL a scriptului.

Utilizarea unei imagini pentru expedierea datelor

Aspectul unui buton de expediere este monoton și obișnuit. Dacă aspectul paginii dumneavoastră este important, puteți folosi o imagine în locul unui buton de expediere. O asemenea imagine se numește, uneori, *buton imagine*. Când un utilizator execută clic pe un buton imagine, datele din formular sunt trimise serverului, ca și cum utilizatorul ar fi executat clic pe un buton de expediere.

Pentru a crea un buton imagine, folosiți următoarea sintaxă:

```
<INPUT TYPE="IMAGE" SRC="url"
NAME="text" ALIGN="alinie">
```

Atributele SRC și TYPE sunt obligatorii; celelalte atribute sunt opționale. Atributul SRC determină adresa URL a fișierului care conține imaginea ce va fi afișată. Atributul NAME atribuie un nume controlului de tip buton imagine. Atributul ALIGN poate lua oricare din valorile top (sus), middle (la mijloc) sau bottom (jos) și specifică modul de aliniere a butonului imagine relativ la textul înconjurător.

Sugestie

Proiectul dat la sfârșitul acestei secțiuni include un buton imagine. Efectuați acest proiect, pentru a vă familiariza cu aspectul și modul de comportare a butoanelor imagine.

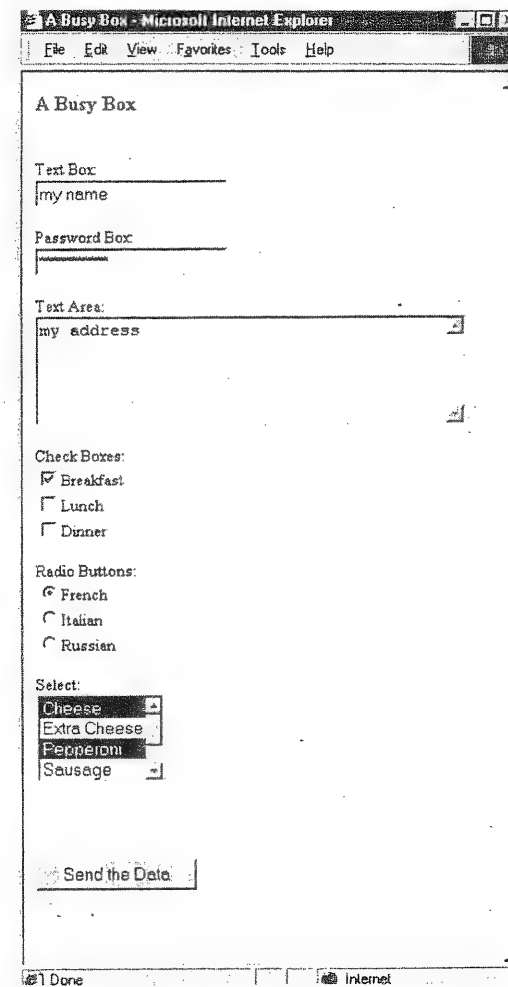


Figura 3-1 O casetă aglomerată care conține un formular HTML ce include controale

Variable	Value
PHP_SELF	/-bmccarty/php/module-03/project-3-2a.php
HTTP_POST_VARS["textbox"]	my name
HTTP_POST_VARS["passwordbox"]	my password
HTTP_POST_VARS["textarea"]	my address
HTTP_POST_VARS["breakfast"]	on
HTTP_POST_VARS["radiobutton"]	french
HTTP_POST_VARS["select"]	array ([0] => cheese [1] => pepperoni)
HTTP_POST_VARS["treasure"]	inestimable

Figura 3-2 Valorile câmpurilor trimise serverului de către caseta aglomerată care conține un formular HTML ce include controale

Crearea unui buton de reinițializare

Uneori, este folositor ca utilizatorul să poată executa clic pe un buton care să ștergă toate informațiile incluse într-un formular. Un asemenea buton se numește *buton de reinițializare*. Pentru a crea un buton de reinițializare, folosiți următoarea sintaxă:

```
<INPUT TYPE="RESET" VALUE="text">
```

Unicul atribut obligatoriu este TYPE. Atributul opțional VALUE specifică textul care va apărea pe suprafața butonului de reinițializare; dacă atributul este omis, pe buton va scrie „Reset“.

Crearea unei pagini care conține mai multe formulare

Așa cum s-a arătat, puteți include mai multe formulare într-o singură pagină HTML. Fiecare formular trebuie să-și aibă propriul său buton sau propria sa imagine pentru expedierea datelor. Un buton de reinițializare din cadrul unui formular se aplică numai formularului care îl conține.

Iată un exemplu de pagină care conține mai multe formulare:

```
<HTML>
<HEAD>
<TITLE>O pagina cu mai multe formulare</TITLE>
</HEAD>
<BODY>
```

```
<!--Primul formular -->
<FORM METHOD="POST"
  ACTION="procesare-formular-client.php">
<BR>Numele clientului:
<BR><INPUT TYPE="TEXT" NAME="nume_client">
<BR><INPUT TYPE="SUBMIT"
  VALUE="Trimite formularul clientului">
</FORM>
<!--Al doilea formular -->
<FORM METHOD="POST"
  ACTION="procesare-formular-furnizor.php">
<BR>Numele furnizorului:
<BR><INPUT TYPE="TEXT" NAME="nume_furnizor">
<BR><INPUT TYPE="SUBMIT"
  VALUE="Trimite formularul furnizorului">
</FORM>
</BODY>
</HTML>
```

Pagina conține două formulare: unul pentru expedierea informațiilor aferente clientului și altul pentru expedierea informațiilor referitoare la furnizor. Fiecare formular își are propriul său buton de expediere. În funcție de butonul de expediere pe care s-a executat clic, la server va fi trimis, în vederea prelucrării, numele unui client sau al unui furnizor.

În acest exemplu, fiecare formular dispune de propriul său script de prelucrare. Cu toate acestea, se poate folosi un singur script pentru prelucrarea datelor provenite de la fiecare formular. Un asemenea script poate determina dacă lucrează cu un formular de client sau cu unul de furnizor, în funcție de câmpurile și valorile pe care le primește.

Utilizarea unei legături pentru a furniza date unui script

În afară de a expedia unui script datele dintr-un formular prin intermediul câmpurilor din formular, puteți expedia date cu ajutorul adresei URL a scriptului, așa cum se specifică în atributul ACTION al etichetei FORM. Pentru aceasta, atașați la sfârșitul adresei URL un semn al întrebării (?) și apoi includeți o serie de perechi nume-valoare cu următoarea formă:

```
nume1=valoare1&nume2=valoare2&nume3=valoare3
```

Exemplul include numai trei perechi nume-valoare; cu toate acestea, puteți include oricâte asemenea perechi doriți, în funcție de limita impusă de browserul utilizatorului.

? Sfatul specialistului

Întrebare: Dacă doresc să trimit unui script, prin intermediul adresei sale URL, caractere speciale precum un semn al întrebării, un semn egal sau un ampersand? Nu se creează confuzie în acest mod?

Răspuns: Da. Pentru a funcționa corect, un șir trebuie să fie *codificat URL*. Pentru a codifica URL un șir care conține o interogare, caracterele speciale se înlocuiesc cu echivalentele lor hexazecimale, precedate de un simbol procent (%). Pentru detalii, consultați documentul Request for Comments (RFC) 1738, disponibil la adresa www.rfc.net. Unele dintre cele comune caractere speciale și echivalentele lor codificate URL sunt prezentate în tabelul 3-1.

De exemplu, forma codificată URL a șirului "la multi ani!" este %22la.multi%21%22.

Adresa URL rezultantă se numește *șir de interogare* și nu poate conține spații. Dacă doriți să trimiteți un spațiu ca parte a unui șir de interogare, trimiteți în locul spațiului un semn plus (+). Iată un exemplu de șir de interogare care codifică numele autorului acestei cărți:

<http://www.osborne.com/search.cgi?author=Bill+McCarty>

Tabelul 3-1 Codificările URL ale caracterelor speciale frecvent utilizate

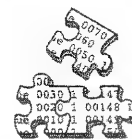
Caracter special	Echivalent codificat URL
.	%2e
?	%3e
>	%5e
~	%7e
+	%2b
.	%2c
/	%2f
:	%3a
;	%3b
<	%3c
=	%3d
>	%3e
[%5b
\	%5c
]	%5d
-	%5f
{	%7b
	%7c
}	%7d
Tab	%09
Spațiu	%20
!	%21

Caracter special	Echivalent codificat URL
"	%22
#	%23
\$	%24
%	%25
&	%26
'	%27
(%28
)	%29
@	%40
'	%60



Test „la minut”

- Scrieți un program HTML care creează un buton imagine cu numele „Start!”
- Scrieți un program HTML care creează un buton de inițializare cu numele „Reluare de la început”.
- Scrieți un șir de interogare care trimite variabila culoare și valoarea roșu serverului de la adresa www.osborne.com/test.



Proiect 3-3: O pagină HTML care conține mai multe formulare

În cadrul acestui proiect, veți crea o pagină HTML care conține două formulare. Un formular are un buton de expediere, iar celălalt are un buton imagine. Ambele formulare dispun de butoane de reinițializare. Valorile atributelor ACTION ale celor două formulare includ șiruri de interogare codificate URL trimise serverului în momentul expedierii formularului.

Scopurile proiectului

- Dobândirea de experiență în crearea și utilizarea paginilor care conțin mai multe formulare
- Prezentarea aspectului și a comportamentului butoanelor imagine și a butoanelor de reinițializare
- Prezentarea șirurilor de interogare

Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit p-3-3.php și încărcați acest fișier pe serverul dumneavoastră PHP:

```
<?php
// Fișierul p-3-3.php
phpinfo()
?>
```

Răspunsuri la test:

- `<INPUT TYPE="IMAGE" SRC="xxx" NAME="Start!">`
- `<INPUT TYPE="RESET" VALUE="Reluare de la început">`
- <http://www.osborne.com/test/?culoare=rosu>

2. Plasați următoarea pagină HTML într-un fișier denumit p-3-3.html și încărcați acest fișier în serverul dumneavoastră PHP, plasându-l în același catalog ca și fișierul p-3-2.php:

```
<HTML>
<HEAD>
<TITLE>Alte detalii despre expedierea formularelor</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-3-3.html -->
<H3>Alte detalii despre expedierea formularelor</H3>
<FORM METHOD="POST"
  ACTION="p-3-3.php?module=3&project=3">
  <BR>Numele clientului:
  <BR><INPUT TYPE="TEXT" NAME="numele_clientului">
  <BR>
  <BR><INPUT TYPE="RESET" VALUE="Sterge datele din formular">
  <BR>
  <BR><INPUT TYPE="SUBMIT" VALUE="Trimite formularul">
</FORM>
<HR>
<FORM METHOD="POST" ACTION="p-3-3.php">
  <BR>Numele furnizorului:
  <BR><INPUT TYPE="TEXT" NAME="numele_furnizorului">
  <BR>
  <BR><INPUT TYPE="RESET" VALUE="Sterge formularul">
  <BR>
  <BR><INPUT TYPE="IMAGE" SRC="button.gif">
</FORM>
</BODY>
</HTML>
```

Apoi, descărcați fișierul button.gif din situl Web aferent acestei cărți și încărcați-l în serverul dumneavoastră PHP, plasându-l în același catalog ca și scriptul PHP și fișierul HTML.

3. Orientați un browser Web spre adresa URL a fișierului care conține formularul HTML. Browserul dumneavoastră va afișa date asemănătoare celor prezentate în figura 3-3. Completați ambele câmpuri text și executați clic pe unul din butoanele de reinițializare (Clear the Form). Remarcați că butonul de reinițializare șterge numai câmpurile din formularul care conține butonul respectiv.
4. Executați clic pe butonul de expediere. Browserul dumneavoastră trebuie să afișeze ceva similar cu ilustrația prezentată în figura 3-4.
5. Observați modul în care perechile nume-valoare din șirul de interogare apar în lista variabilelor PHP, alături de valoarea cuprinsă în caseta cu text.
6. Folosiți butonul Back al browserului pentru a reveni la pagina cu mai multe formulare. Executați clic pe butonul imagine. Rețineți diferențele dintre ilustrația afișată de browser și cea prezentată în figura 3-4. Remarcați modul în care un script PHP poate executa prelucrarea mai multor formulare, prin examinarea variabilelor și a valorilor acestora?

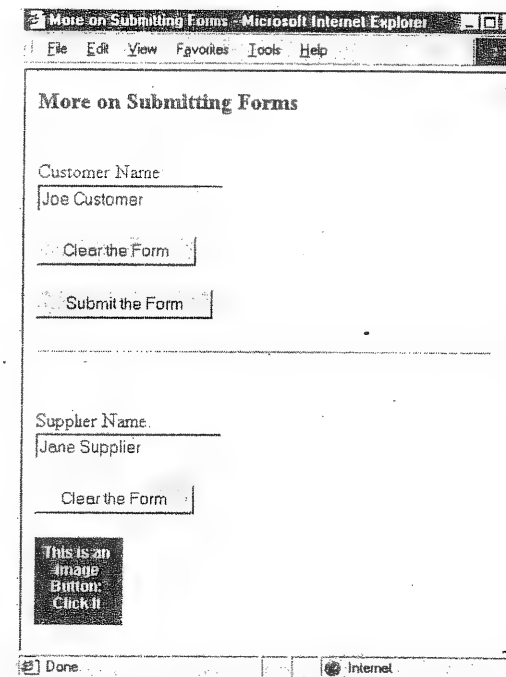


Figura 3-3 Pagina care conține mai multe formulare

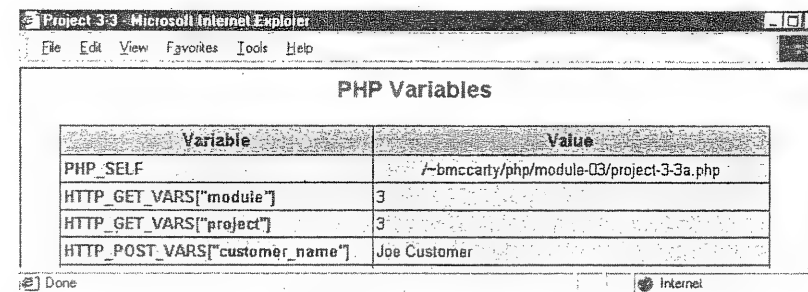


Figura 3-4 Valorile câmpurilor trimise serverului de către pagina care conține mai multe formulare.

Test de evaluare

1. Scrieți o etichetă HTML FORM care își trimite datele unui script situat la adresa `http://www.osborne.com/cgi-bin/test`.
2. Scrieți un program HTML care creează un control cu mai multe linii, denumit adresa, pentru introducerea de text. Controlul trebuie să poată conține 5 rânduri a 80 de caractere fiecare.
3. Scrieți un program HTML care creează un meniu derulant denumit culoare, care conține principalele culori substructive, în speță roșu, galben și albastru. Faceți de așa manieră încât meniul să accepte o singură selecție. Specificați culoarea roșie ca opțiune prestabilită.
4. Scrieți un program HTML pentru crearea unui set de butoane radio denumite dimensiune, care permit utilizatorului să aleagă din următoarele valori: mic, mediu și mare.
5. Scrieți un program HTML pentru crearea unui formular care își trimite datele la adresa `www.dev.null`. Formularul trebuie să conțină un câmp ascuns denumit script, care conține meniul cu valori.

Modulul 4: Accesul la date

Scopuri

- Învățați să obțineți accesul la datele scalare transmise unui program PHP de către un formular HTML
- Învățați să folosiți construcția PHP `echo` pentru a trimite date de ieșire la browser
- Învățați să construiți șiruri care includ valorile variabilelor PHP
- Învățați să obțineți accesul la valorile variabilelor de mediu

Majoritatea programelor PHP folosesc formularele HTML pentru a obține date de intrare. În cadrul acestui modul, veți învăța să obțineți accesul la datele trimise unui program PHP prin intermediul unui formular HTML. De asemenea, veți învăța să obțineți accesul la datele stocate în variabilele de mediu pe parte de client, respectiv pe parte de server. Variabilele de mediu stochează informații utile despre browsere, servere Web și PHP.

Obținerea și utilizarea datelor de la un formular

Deoarece PHP a fost conceput ca limbaj de scripting pentru Web, facilitează accesul la variabilele transmise de către formularele HTML. În modulul precedent, ați învățat să creați formulare HTML. Iată un formular HTML simplu, care include o casetă cu text:

```
<HTML>
<HEAD><TITLE>Un formular HTML simplu</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="phpinfo.php">
<BR>Tastati niste date:
<BR><INPUT TYPE="TEXT" NAME="date">
<BR><BR><INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

Observați că atributul `NAME` al etichetei `INPUT` atribuie casetei cu text numele `date`.

În cazul în care creați un script denumit `phpinfo.php`, care execută funcția `phpinfo()` și o stochează în același dosar ca și formularul, prin expedierea formularului se cere scriptului să afișeze un raport care indică starea serverului de aplicație PHP. În

secțiunea intitulată „Variabile PHP“, raportul de stare prezintă valorile variabilelor din formular. Figura 4-1 prezintă raportul de stare asociat formularului HTML simplu.

Rețineți că raportul de stare are două coloane. Numele variabilei asociate controlului din formular, în speță date, apare în coloana din stânga, înglobat în textul `HTTP_POST_VARS["date"]`. Valoarea variabilei, care reprezintă textul introdus de utilizator, este prezentată în coloana din dreapta. În figură, valoarea variabilei o constituie textul „acestea sunt datele“.

Variable	Value
PHP_SELF	/-bmccarty/php/module-04/phpinfo.php
HTTP_POST_VARS["date"]	this is data

Figura 4-1 Un raport de stare PHP, care indică valoarea variabilei din formular denumită date

? Sfatul specialistului

Întrebare: Construcția `HTTP_POST_VARS["date"]` este suspect de asemănătoare cu o referință la un tablou. Despre ce este vorba?

Răspuns: Dacă ați observat parantezele drepte și v-ați reamintit că ele sunt asociate cu tablourile PHP, atunci sunteți o persoană extrem de atentă. Dacă nu, parcurgeți rapid secțiunea intitulată „Valori scalare și tablouri“ din Modulul 2.

În momentul expedierii unui formular, numele și valorile variabilelor incluse în formular sunt inserate în tabloul `HTTP_POST_VARS`. Cheia fiecărui element al tabloului este numele unei variabile din formular; valoarea elementului din tablou determină valoarea variabilei. Dacă doriți, puteți obține accesul la variabilele formularului folosind tabloul `HTTP_POST_VARS`; cu toate acestea, veți învăța în curând modalități mult mai convenabile de acces la variabilele unui formular.

Trimiterea datelor de ieșire către browser

Desigur, nu putem pretinde ca utilizatorii să citească un raport de stare PHP pentru a afla valorile variabilelor. Mai mult, esența programării constă în capacitatea de a manipula valorile, nu de a le vizualiza pur și simplu. În această sub-secțiune veți învăța să folosiți construcția `echo` pentru a trimite date de ieșire la un browser, iar în secțiunea următoare veți învăța să construiți șiruri care înglobează valorile variabilelor.

Prin combinarea acestor tehnici, puteți afișa valorile variabilelor PHP într-o formă pe care utilizatorii o pot înțelege cu ușurință. În modulele ulterioare, veți învăța să manipulați valorile variabilelor astfel încât să puteți construi programe PHP utile.

Construcția `echo` vă permite să trimiteți date de ieșire către un browser. Construcția are o diversitate de forme. Iată-o, poate, pe cea mai utilă dintre ele:

```
echo sir_expresie;
```

Această formă constă numai din cuvântul `echo`, urmat de o expresie șir și de un caracter punct și virgulă care determină încheierea instrucțiunii. Efectul unei asemenea instrucțiuni constă în a trimite browserului valoarea expresiei șir. De exemplu, instrucțiunea

```
echo "<BR><B>Datele sunt elegante!</B>";
```

trimite browserului textul „Datele sunt elegante!“. Rețineți că textul include etichete HTML, care influențează modul de formatare a textelor, determinându-le să apară pe o linie nouă, scrise cu ajutorul caracterelor aldine.

Utilitatea acestei forme a construcției `echo` se bazează pe numeroasele moduri în care puteți scrie o expresie șir. Una dintre cele mai utile modalități folosește operatorul de concatenare pentru unirea a două expresii șir. De exemplu, să luăm în considerare următoarea instrucțiune `echo`:

```
echo "Datele sunt " . "elegante!";
```

Operatorul de concatenare, reprezentat printr-un punct, atașează a doua expresie șir la prima. Rezultatul este același ca și cel generat de următoarea instrucțiune:

```
echo "Datele sunt elegante!";
```

Construcția `echo` este oarecum ciudată, în sensul că este asemănătoare cu o funcție, dar nu este funcție. De exemplu, puteți folosi paranteze pentru a delimita expresia șir cerută de construcția `echo`:

```
echo("Datele sunt elegante!");
```

Dacă preferați, puteți furniza mai multe expresii șir, fiecare din aceste expresii fiind separată de vecinii săi prin intermediul unei virgule:

```
echo "Datele sunt", "elegante!";
```

Cu toate acestea, nu puteți folosi simultan paranteze și virgule, așa cum se procedează în cazul funcțiilor autentice:

```
echo("Datele sunt", "elegante!"); // Eroare de sintaxă
```

Mai mult, o expresie furnizată construcției `echo` nu trebuie să fie o expresie șir, dacă PHP știe să convertească expresia într-un șir. De exemplu, următoarea sintaxă este corectă:

```
echo "unu plus unu este "; echo 2;
```

? Sfatul specialistului

Întrebare: Să presupunem că programul meu PHP are ca date de ieșire etichete HTML, ceea ce determină intercalarea unor elemente HTML în componentele programului meu. Există vreo modalitate mai convenabilă de a scrie construcții de tip echo în această situație?

Răspuns: Da, există. În loc de a scrie

```
<?
echo "datele mele de iesire";
?>
```

puteți scrie pur și simplu

```
<?= "datele mele de iesire" ?>
```

PHP percepe semnul egal ca o abreviere pentru echo, atâta vreme cât semnul egal urmează imediat după simbolul <?. Rețineți, totuși, că această caracteristică este disponibilă numai dacă PHP a fost compilat sau configurat cu opțiunea de configurare short_tags. Dacă descoperiți că nu puteți folosi cu succes construcția <?=:, solicitați administratorului dumneavoastră de sistem să activeze această opțiune.

Construirea șirurilor care înglobează valori ale variabilelor

Pentru a putea trimite browserului valorile variabilelor, tot ce mai aveți de învățat este să construiți expresii șir care includ valorile variabilelor. Iată un script simplu care include o asemenea expresie:

```
<?php
$raza = 2;
$pi = 3.14159;
$arie = $pi * $raza * $raza;
echo "Aria cercului este $arie.";
?>
```

Când înglobați într-un șir numele unei variabile, precum \$arie, PHP înlocuiește numele variabilei cu valoarea acesteia. Dacă executați scriptul dat ca exemplu, veți vedea datele de ieșire:

```
Aria cercului este 12.56636.
```

Uneori, doriți să obțineți la ieșire numele unei variabile, nu valoarea acesteia. În asemenea cazuri, inserați un caracter backslash (\) în fața numelui variabilei. Să considerăm următorul exemplu:

```
<?php
$raza = 2;
$pi = 3.14159;
```

```
$arie = $pi * $raza * $raza;
echo "Valoarea variabilei \$arie este $arie.";
?>
```

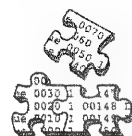
Rezultatul acestui script este:

```
Valoarea variabilei $arie este 12.56636.
```



Test „la minut”

- Scrieți o instrucțiune echo care scrie textul „PHP este pentru programatorii de calibru.”
- Scrieți o instrucțiune echo care scrie valoarea variabilei \$circum.
- Scrieți o instrucțiune echo care scrie numele variabilei \$radacina, urmată de un semn egal și de valoarea variabilei.



Proiectul 4-1: Agenda cu adrese de e-mail

În cadrul acestui proiect, veți scrie instrucțiuni PHP prin care se obține accesul la datele obținute de la un formular HTML. De asemenea, veți scrie instrucțiuni PHP care trimit date HTML la browserul utilizatorului.

Acest proiect este primul dintr-o serie de proiecte, care va culmina cu o aplicație PHP care furnizează o agendă de adrese accesibilă prin Web. Veți învăța mai multe despre agenda de adrese și despre funcționalitățile acesteia pe măsură ce veți continua să lucrați la dezvoltarea aplicației.

Scopurile proiectului

- Prezentarea modului de scriere a șirurilor PHP care încorporează datele obținute de la un formular HTML
- Prezentarea modului de utilizare a instrucțiunii PHP echo pentru a trimite date către browserul utilizatorului

Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit p-4-1.html și încărcați acest fișier în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Proiectul 4-1</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-4-1.html -->
<FORM METHOD="POST" ACTION="p-4-1.php">
<H2>Lista cu persoane de contact</H2>
<BR>Porecla:
```

Răspunsuri la test:

- echo "PHP este pentru programatorii de calibru."
- echo "Valoarea este \$circum."; sau ceva similar
- echo "\\$radacina=\$radacina";


```

<BR><INPUT TYPE="TEXT" NAME="Porecla">
<BR>
<BR>Nume complet:
<BR><INPUT TYPE="TEXT" NAME="Numecomplet">
<BR>
<BR>Memo:
<BR><TEXTAREA NAME="Memo" ROWS="4" COLS="40" WRAP="PHYSICAL">
</TEXTAREA>
<BR>
<BR>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>

```

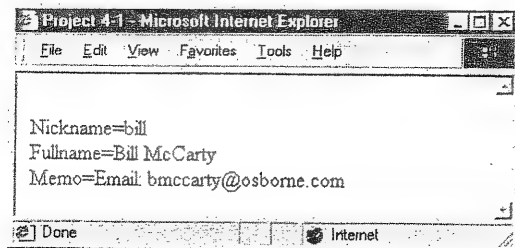
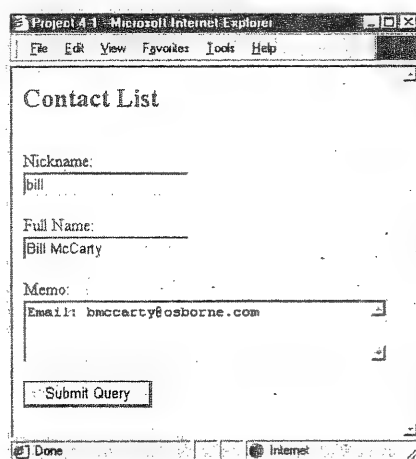
2. Plasați următorul script PHP într-un fișier denumit p-4-1.php și încărcați acest fișier în serverul dumneavoastră PHP, plasându-l în același catalog ca și fișierul p-4-1.html:

```

<?php
//Fișierul p-4-1.html
echo "<BR>Porecla=$Porecla";
echo "<BR>Nume_complet=$Nume_complet";
echo "<BR>Memo=$Memo";
?>

```

3. Orientați un browser Web spre adresa URL a fișierului care conține formularul HTML. Ecranul browserului trebuie să fie asemănător celui prezentat în ilustrația următoare. Introduceți o poreclă, un nume complet și o notă scurtă (memo). Executați clic pe butonul de expediere.
4. În momentul executării scriptului de prelucrare, acesta obține accesul la cele trei variabile ale formularului și trimite browserului utilizatorului numele și valorile variabilelor, așa cum se poate vedea în ilustrația următoare.



Obținerea și utilizarea datelor de la o variabilă de mediu

În cazul în care sunteți familiarizat cu sistemele de operare UNIX sau MS-DOS, probabil că sunteți un cunoscător al variabilelor de mediu. Variabilele de mediu sunt folosite pentru stocarea opțiunilor și a parametrilor care personalizează mediul de aplicație. Aplicațiile pot obține accesul la valorile variabilelor de mediu și își pot ajusta comportamentul în consecință. De exemplu, calea de căutare a programelor MS-DOS este stocată într-o variabilă de mediu denumită PATH. În general, comenzile sistemelor de operare sunt folosite pentru a configura variabilele de mediu și pentru a stabili valorile acestora. Cu toate acestea, unele aplicații manipulează valorile variabilelor de mediu.

Atât serverul Web Apache, cât și serverul de aplicație PHP folosesc variabile de mediu pentru a prezenta informații de stare. Unele dintre cele mai importante variabile de mediu folosite de Apache și PHP sunt rezumate în tabelul 4-1. Numeroase servere Web, altele decât Apache, furnizează o parte din aceste variabile de mediu sau chiar pe toate.

Multe dintre aceste variabile reflectă caracteristicile cererii HTTP care a solicitat execuția PHP. Puteți vizualiza toate variabilele de mediu disponibile pentru programele PHP prin invocarea funcției `phpinfo()` și vizualizarea datelor de ieșire generate de aceasta. Figura 4-2 prezintă o porțiune a raportului de stare prezentat de funcția `phpinfo()` care identifică numeroase variabile de mediu.

Tabelul 4-1 Importante variabile de mediu PHP

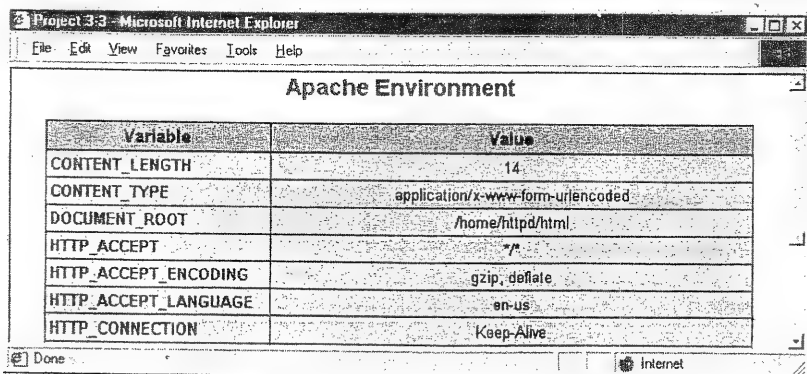
Variabilă de mediu	Descriere
CONTENT_LENGTH	Lungimea, în octeți, a corpului cererii.
CONTENT_TYPE	Tipul MIME al datelor din corpul cererii.
DOCUMENT_ROOT	Calea care constituie rădăcina arborelui catalogului cu documente al serverului Web.
GATEWAY_INTERFACE	Versiune a protocolului CGI (Common Gateway Interface) folosit de serverul Web.
HTTP_ACCEPT	Conținutul antetului HTTP Accept:.
HTTP_ACCEPT_CHARSET	Conținutul antetului HTTP Accept-Charset:, care specifică seturile de caractere înțelese de client.
HTTP_ACCEPT_ENCODING	Conținutul antetului HTTP Accept-Encoding:, care specifică tipurile de conținuturi înțelese de client.
HTTP_ACCEPT_LANGUAGE	Conținutul antetului HTTP Accept-Language:, care specifică limbajele preferate de client.
HTTP_CONNECTION	Conținutul antetului HTTP Connection:, care indică opțiunile solicitate de client.
HTTP_HOST	Conținutul antetului HTTP Host:, care indică numele de gazdă, folosit de client la prezentarea cererii.
HTTP_REFERER	Adresa URL a paginii Web care a trimis browserul clientului la pagina curentă.

Tabelul 4-1 Continuare

Variabilă de mediu	Descriere
HTTP_USER_AGENT	Conținutul antetului HTTP User-Agent, care indică browserul clientului și versiunea acestuia.
PATH	Calea de execuție asociată cu mediul serverului.
QUERY_STRING	Șirul de interogare, dacă există, care a însoțit cererea.
REMOTE_ADDR	Adresa IP a clientului.
REMOTE_HOST	Numele de gazdă al clientului.
REMOTE_PORT	Adresa portului clientului de unde a pornit cererea.
REQUEST_METHOD	Metoda de cerere HTTP folosită; de exemplu, GET, POST, PUT sau HEAD.
REQUEST_URI	URI folosit pentru accesul la pagina curentă. URI este alcătuit dintr-un URL și un șir opțional de interogare.
SCRIPT_FILENAME	Numele de cale absolut al scriptului curent.
SCRIPT_NAME	Adresa URL a scriptului curent.
SERVER_ADMIN	Adresa de e-mail a administratorului serverului Web.
SERVER_HOST	Numele de gazdă asociat serverului Web care prelucrează cererea.
SERVER_PORT	Port folosit de serverul Web pentru comunicații.
SERVER_PROTOCOL	Numele și versiunea protocolului prin intermediul căruia s-a executat cererea.
SERVER_SIGNATURE	Șirul care identifică versiunea serverului Web și numele de gazdă folosit pentru prelucrarea cererii.
SERVER_SOFTWARE	Șirul care identifică programul server Web și versiunea acestuia.

Puteți obține accesul la variabila de mediu exact așa cum procedați pentru orice altă variabilă PHP. Pur și simplu inserați înaintea numelui variabilei de mediu un simbol al dolarului (\$), astfel încât numele să se conformeze sintaxei PHP. De exemplu, următoarea instrucțiune echo trimite browserului adresa IP a clientului:

```
echo "Adresa IP a clientului este $REMOTE_ADDR."
```



Variable	Value
CONTENT_LENGTH	14
CONTENT_TYPE	application/x-www-form-urlencoded
DOCUMENT_ROOT	/home/httpd/html
HTTP_ACCEPT	*/*
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_ACCEPT_LANGUAGE	en-us
HTTP_CONNECTION	Keep-Alive

Figura 4-2 Datele de ieșire ale funcției phpinfo(), care afișează numeroase variabile de mediu.



Sfatul specialistului

Întrebare: O bună parte din informațiile prezentate în tabelul 4-1 par neclare. Care este utilitatea acestor variabile de mediu?

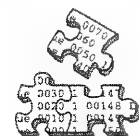
Răspuns: Dacă nu cunoașteți protocolul HTTP în amănunțime, s-ar putea să nu descoperiți prea multe utilități pentru aceste variabile de mediu. Cu toate acestea, o importantă utilizare comună o constituie autentificarea clientului. Prin accesul la variabila de mediu REMOTE_ADDR, puteți determina adresa IP a clientului. Într-un modul ulterior, veți învăța să testați valoarea unei variabile și să modificați comportamentul unui script în funcție de valoarea respectivă. De exemplu, puteți folosi acest procedeu pentru a exclude cererile care provin din afara unui anumit set de adrese IP, cum ar fi cele care reprezintă o anumită rețea. Astfel, aplicația dumneavoastră poate deveni mai rezistentă la atacurile hackerilor care încearcă să creeze o breșă în sistemul de securitate.

O altă utilizare importantă a variabilelor de mediu constă în ocolirea limitărilor impuse de un anumit browser. Variabila de mediu HTTP_USER_AGENT identifică browserul client și versiunea acestuia. Un script PHP poate testa valoarea acestei variabile de mediu și trimite numai date de ieșire acceptabile pentru versiunea browserului aflat în uz.



Test „la minut”

- Care este numele variabilei PHP care conține numele gazdei serverului Web?
- Care este numele variabilei PHP care conține numele gazdei clientului?



Proiect 4-2: Vizualizarea variabilelor de mediu

În cadrul acestui proiect, veți vizualiza valorile a numeroase variabile de mediu PHP.

Scopurile proiectului

- Prezentarea modului de vizualizare a variabilelor de mediu
- Prezentarea modului de utilizare a instrucțiunii echo pentru a trimite browserului date de ieșire

Pas cu pas

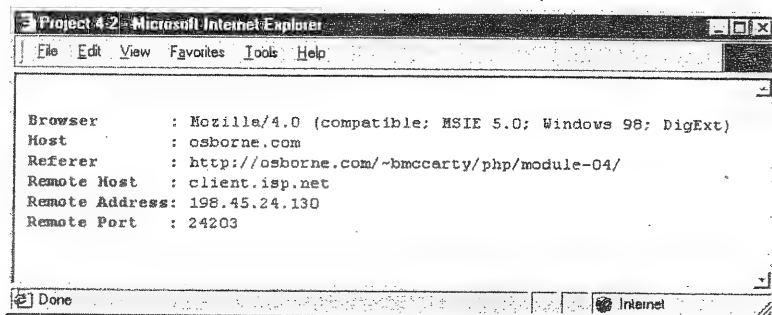
1. Plasați următorul script PHP într-un fișier denumit p-4-2.php și încărcați acest fișier în serverul dumneavoastră PHP:

Răspunsuri la test:

- \$SERVER_HOST
- \$REMOTE_HOST


```
<?php
// Fișierul p-4-2.php
echo "<PRE>";
echo "<BR><B>Browser      :</B> $HTTP_USER_AGENT";
echo "<BR><B>Host          :</B> $HTTP_HOST";
echo "<BR><B>Referer       :</B> $HTTP_REFERER";
echo "<BR><B>Remote Host   :</B> $REMOTE_HOST";
echo "<BR><B>Remote Address:</B> $REMOTE_ADDR";
echo "<BR><B>Remote Port   :</B> $REMOTE_PORT";
echo "</PRE>";
?>
```

2. Orientați un browser Web spre adresa URL a fișierului care conține scriptul PHP. La executarea scriptului de prelucrare, acesta afișează valorile variabilelor de mediu, așa cum se poate vedea în ilustrația următoare.



Test de evaluare

1. Care este variabila PHP ce trebuie folosită pentru a obține accesul la datele asociate unui control definit de eticheta HTML `<INPUT TYPE="TEXT" NAME="culoare">`?
2. Scrieți o instrucțiune PHP care trimite browserului valoarea variabilei `$x`.
3. Scrieți o instrucțiune PHP care trimite browserului numele variabilei `$y`.
4. Scrieți o instrucțiune PHP care trimite browserului adresa URL a paginii care face referire la pagina curentă.

Modulul 5: Lucrul cu valori scalare

Scopuri

- Învățați să definiți și să utilizați constantele
- Învățați să folosiți variabilele dinamice
- Învățați să convertiți valorile dintr-un tip în altul

În cadrul Modulului 2, ați învățat despre valori scalare și despre tablouri. În timp ce un tablou are mai multe valori asociate, un scalar are asociată o singură valoare. În acest modul, veți învăța mai multe despre valorile scalare și despre modul de utilizare a acestora.

Utilizarea constantelor și a variabilelor dinamice

În sub-secțiunile următoare, vom discuta despre constante și variabile dinamice. Puteți scrie programe PHP utile și complexe fără a utiliza constante sau variabile dinamice. Dacă utilizarea constantelor poate facilita citirea programelor dumneavoastră, variabilele dinamice au un efect *contrar*. Din acest motiv, în general se recomandă evitarea variabilelor dinamice, mai ales de către programatorii PHP începători. Totuși, chiar dacă optați pentru a nu folosi variabilele dinamice, vă puteți afla în situația de a lucra la un program PHP scris de o persoană care agreează aceste variabile; ca atare, trebuie să aveți cunoștințe despre variabilele dinamice, indiferent dacă le folosiți sau nu în propriile dumneavoastră programe.

Utilizarea constantelor

O constantă este pur și simplu o valoare care este... constantă, cu alte cuvinte o valoare care nu se modifică. În acest sens, constantele sunt opusele variabilelor, deoarece valoarea unei variabile se poate modifica pe durata execuției unui program. Dacă preferați, gândiți-vă la o constantă ca la o variabilă „numai pentru citire”.

Pentru a defini o constantă, folosiți funcția `define()`. Să considerăm următorul exemplu:

```
define("PI", 3.14159);
```

Această instrucțiune definește constanta `PI`, atribuindu-i valoarea 3.14159. După ce a fost definită, o constantă se poate folosi în cadrul unei expresii. De exemplu, puteți calcula aria unui cerc după cum urmează:

```
$arie = PI * $raza * $raza;
```

Observați că referințele la o constantă nu folosesc simbolul dolarului. Astfel, o constantă poate fi cu ușurință deosebită de o variabilă. Mulți programatori scriu numele constantelor folosind numai majuscule, ceea ce le face și mai simplu de identificat.

O funcție conexă, `defined()`, poate determina dacă o anumită constantă a fost definită. De exemplu, cu ajutorul următoarei instrucțiuni PHP puteți determina dacă a fost definită constanta `PI`:

```
echo defined("PI");
```

Rețineți că numele care va fi testat este delimitat prin ghilimele duble. Funcția `defined()` returnează o valoare unitară dacă respectiva constantă a fost specificată; în caz contrar, returnează zero. În cadrul exemplului, instrucțiunea `echo` afișează valoarea returnată. Puteți testa valoarea returnată și cu ajutorul construcțiilor PHP descrise în Modulul 6.

Pe lângă sporirea lizibilității programelor, constantele pot facilita modificarea acestora. Să presupunem că ați scris un program care conține multe calcule ce folosesc valoarea 3,14159, iar ulterior ați descoperit că trebuia să folosiți valoarea mai exactă 3,1415926535898. Descoperirea și modificarea fiecărei apariții a valorii originale poate fi o activitate mare consumatoare de timp. Dar, dacă ați definit o constantă pentru reprezentarea valorii, numărul 3,14159 va apărea o singură dată în program. Modificarea unei singure apariții va deveni o operație simplă și obișnuită. Unii programatori cred că expresiile trebuie să conțină numai două valori numerice: zero și unu. Toate celelalte valori trebuie reprezentate sub formă de constante, pentru a îmbunătăți lizibilitatea și a facilita modificarea programelor.

Valoarea π este folosită extrem de frecvent în unele calcule. Pentru comoditate, PHP furnizează o funcționalitate mai indicată decât definirea unei constante cu valoarea π : funcția `pi()` returnează valoarea respectivă, cu 14 cifre semnificative. Deci, puteți calcula aria unui cerc folosind următoarea expresie:

```
$arie = pi() * $raza * $raza;
```

Lucrul cu variabile dinamice

Dacă o constantă poate spori lizibilitatea și simplitatea modificării programelor, variabilele dinamice îngreunează înțelegerea și posibilitatea de a opera schimbări în program. Iată un exemplu simplu de variabilă dinamică, denumită `$$film`:

```
$oameni_furiosi = 12;
$$film = "oameni_furiosi";
echo $$film;
```

O variabilă dinamică este denumită folosind o pereche de simboluri ale dolarului și este asociată cu o variabilă obișnuită care are un nume similar și include un singur

simbol al dolarului. În exemplu, variabila dinamică `$$film` este asociată cu variabila obișnuită `$film`. Valoarea unei variabile obișnuite dă numele (fără un simbol al dolarului) unei a doua variabile ordinare. În exemplu, a doua variabilă obișnuită este `$oameni_furiosi`. Valoarea acestei a doua variabile obișnuite este valoarea variabilei dinamice: în exemplu, aceasta este valoarea 12*.

Programatorii spun că o variabilă dinamică face o *referință indirectă* la valoarea sa. Cu alte cuvinte, o variabilă dinamică nu conține, practic, propria sa valoare. În schimb, conține amplasamentul unde se poate găsi valoarea; cu alte cuvinte, numele unei alte variabile.



Test „la minut”

- Scrieți o instrucțiune care definește o constantă denumită `LATURI`, atribuindu-i valoarea 4.
- Scrieți o instrucțiune care definește o constantă denumită `SALUT`, atribuindu-i valoarea `"buna_ziua"`.
- Scrieți o instrucțiune care afișează valoarea variabilei dinamice asociate variabilei obișnuite `$porumbel`.



Sfatul specialistului

Întrebare: Variabilele dinamice sunt o noțiune cam derutantă. De ce le folosesc programatorii?

Răspuns: Dacă variabilele dinamice vi se par derutante, nu intrați în panică; variabilele dinamice *sunt* derutante. Uneori este posibilă reducerea dimensiunilor unui program folosind una sau mai multe variabile dinamice. Dar programatorii comit adesea greșeli care determină variabilele dinamice să facă referire la valori greșite și neașteptate. Gândiți-vă că a crea un program corect este mai important decât a crea un program succint. Când vedeți o posibilitate de a elimina numeroase linii de program folosind o variabilă dinamică, gândiți-vă de două ori înainte de a revizui programul.

* În original era vorba despre filmul „39 de trepte”. S-a preferat modificarea exemplului pentru a nu denumi o variabilă cu numele „de_trepte”, așa cum o cerea exemplul. – N.T.
Răspunsuri la test:

- `define("LATURI", 4);`
- `define("SALUT", "buna_ziua");`
- `echo $$porumbel;`

Lucrul cu tipuri

PHP este ceea ce se numește un limbaj de programare cu tipuri dinamice. O consecință a caracterului dinamic al tipurilor de variabile aferente limbajului PHP este aceea că nu trebuie să specificați tipul variabilelor. PHP determină tipul variabilei în funcție de tipul ultimei valori atribuite variabilei.

Cu toate acestea, caracterul dinamic al tipurilor nu vă scutește de problemele legate de tipuri. Trebuie să cunoașteți tipurile acceptate și ceea ce se întâmplă când în cadrul expresiilor se folosesc două sau mai multe tipuri.

Conversia automată de tip

Să luăm în considerare următorul script PHP scurt:

```
$x = 1;
$y = 2.5;
$z = $x + $y;
echo $z;
```

Instrucțiunea de atribuire care stochează o valoare în variabila \$z este interesantă, deoarece expresia din membrul drept include un operand întreg și un operand de tip dublu. Ce valoare va apărea ca dată de ieșire? Răspunsul corect este 3.5, o valoare de tip dublu.

Când o expresie aritmetică folosește mai multe tipuri, PHP execută conversia automată de tip. Dacă oricare dintre operanzi este de tip dublu, PHP tratează ceilalți operanzi ca și cum ar fi de tip dublu, execută calculele și returnează rezultatul ca valoare de tip dublu. Dacă toți operanzii unei expresii sunt întregi, PHP execută calculul și returnează rezultatul sub formă de întreg.

Este important să înțelegeți faptul că prin conversia de tip nu se modifică tipurile operanzilor unei expresii; aceștia sunt pur și simplu tratați ca și cum ar fi fost de un alt tip. În cadrul exemplului, variabila \$x rămâne de tip întreg, chiar dacă PHP o tratează ca o valoare de tip dublu pentru a executa calculele.

Șirurile pot fi de asemenea supuse unei conversii de tip. Să examinăm următorul exemplu:

```
$x = 1;
$y = $x + "1 more";
echo $y;
```

Cuvântul more din șirul text este ignorat

Valoarea afișată este doi, adică suma dintre valoarea variabilei \$x și valoarea numerică a șirului text, care este unu. Valoarea numerică și tipurile unui șir sunt determinate prin respectarea următoarelor reguli:

* În traducere „încă 1”. Cuvântul „more” din șirul text este ignorat. Dacă s-ar fi folosit traducerea în limba română, valoarea variabilei \$y ar fi fost unu, din motive care vor fi prezentate imediat (regula nr. 1). – N.T.

1. Dacă șirul începe cu o valoare numerică, valoarea șirului este dată de valoarea numerică respectivă; în caz contrar, valoarea șirului este zero.
2. Dacă un punct zecimal sau un exponent (e sau E), este asociat cu valoarea numerică, tipul variabilei rezultante este dublu; în caz contrar, tipul valorii rezultante este un întreg.

Conversia manuală de tip

Dacă preferați, puteți prelua controlul conversiei de tip sau puteți modifica tipul unei variabile. Pentru a prelua controlul conversiei de tip, puteți *converti forțat* un operand de la un tip la altul, proces cunoscut sub numele de *conversie forțată de tip* sau pur și simplu *conversie forțată*. În continuare, este dat un exemplu de conversie de tip:

```
$x = 1;
$y = 2.5;
$z = $x + (integer) $y;
echo $z;
```

Conversia forțată de tip, și anume (integer), determină tratarea variabilei \$y ca pe un întreg, iar valoarea acesteia devine 2, în loc de 2.5, care este valoarea reală a variabilei \$z. Tabelul următor indică și alte conversii forțate de tip care se pot folosi:

Conversie forțată	Rezultat
(int), (integer)	Conversie forțată la întreg
(real), (double), (float)	Conversie forțată la dublu
(string)	Conversie forțată la șir
(array)	Conversie forțată la tablou
(object)	Conversie forțată la obiect

Numeroase funcții furnizează încă o modalitate de a trata o variabilă ca și cum ar fi de un tip specificat. Tabelul următor prezintă succint aceste funcții. Ca exemplu de utilizare a uneia dintre funcțiile respective, luați în considerare următorul exemplu:

```
$x = 1.5;
$y = intval($x);
echo $x;
echo $y;
```

Valoarea 1.5 este afișată ca valoare a variabilei \$x, iar valoarea 1 este afișată ca valoare a variabilei \$y.

Funcție	Operație
Doubleval()	Tratează argumentul ca fiind de tip dublu.
Intval()	Tratează argumentul ca fiind de tip întreg.
Strval()	Tratează argumentul ca fiind de tip șir.

Nici conversia normală și nici cea forțată nu modifică tipul unui operand. Ambele mecanisme determină tratarea operanzilor ca și cum ar fi de un alt tip. Totuși, modificarea tipului unei variabile este posibilă prin utilizarea funcției `settype()`. Acest procedeu este ilustrat în următorul exemplu:

```
$x = 1.5;
settype($x, "integer");
echo $x;
```

Valoarea afișată a variabilei `$x` este egală cu unitatea, deoarece fracția zecimală se pierde atunci când funcția `settype()` convertește valoarea dublă la o valoare întreagă. Puteți obține același rezultat cu următoarea instrucțiune de atribuire, care folosește o conversie forțată:

```
$x = (integer) $x;
```

Valorile posibile pentru al doilea argument al funcției `settype()`, și anume argumentul care specifică tipul dorit, sunt:

- „integer“
- „double“
- „string“
- „array“
- „object“

O funcție conexă, și anume `gettype()`, returnează un șir care indică tipul variabilei specificate. Scriptul următor afișează șirul „integer“, care indică tipul variabilei `$x`:

```
$x = 1;
echo gettype($x);
```

? Sfatul specialistului

Întrebare: Studiind programele PHP scrise de alții, am observat operatorul `+=`. Care este efectul operatorului respectiv?

Răspuns: Operatorul `+=` nu este decât o modalitate rapidă de a scrie o instrucțiune de atribuire care implică operatorul `+`. Următoarele două instrucțiuni de atribuire, una normală și una „rapidă“, au același efect:

```
$x += 1;
$x = $x + 1;
```

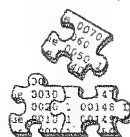
Această instrucțiune de atribuire „rapidă“ vă scutește de efortul de a scrie o variabilă, `$x` în cazul exemplului nostru, în ambii membri ai instrucțiunii de atribuire. Tabelul următor rezumă operatorii „rapizi“ de atribuire pentru operațiile aritmetice și pentru șiruri, furnizând un exemplu pentru fiecare operator și o instrucțiune de atribuire echivalentă.

Instrucțiune de atribuire „rapidă“	Instrucțiune de atribuire normală
<code>x += y</code>	<code>x = x + y</code>
<code>x -= y</code>	<code>x = x - y</code>
<code>x /= y</code>	<code>x = x / y</code>
<code>x *= y</code>	<code>x = x * y</code>
<code>x %= y</code>	<code>x = x % y</code>
<code>x .= y</code>	<code>x = x . y</code>



Test „la minut“

- Dacă se înmulțește o valoare de tip dublu cu o valoare întreagă, care este tipul rezultatului?
- Scrieți o expresie care convertește forțat valoarea variabilei `$x` în tipul dublu.
- Care este valoarea și tipul expresiei `1.5 + "eu"`?



Proiect 5-1: Un calculator simplu

În cadrul acestui proiect, veți scrie și veți executa un mic program PHP care servește drept calculator simplu. Calculatorul adună două numere introduse de utilizator și afișează suma.

Scopurile proiectului

- Prezentarea modului în care numerele introduse sub formă de text se pot folosi în expresiile aritmetice
- Prezentarea modului în care instrucțiunile PHP dintr-un formular pot prelucra datele introduse în formular

Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit `p-5-1.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Proiect 5-1</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-5-1.php -->
<FORM METHOD="POST" ACTION="p-5-1.php">
<TABLE>

<TR>
<TD><INPUT TYPE="TEXT" NAME="OP1"
VALUE="<? echo $OP1 ?>"></TD>
</TR>

<TR>
<TD ALIGN="CENTER">+</TD>
</TR>

<TR>
<TD><INPUT TYPE="TEXT" NAME="OP2"
VALUE="<? echo $OP2 ?>"></TD>
</TR>
```

Răspunsuri la test:

- dublu
- (double) `$x`
- 1.5, dublu

```

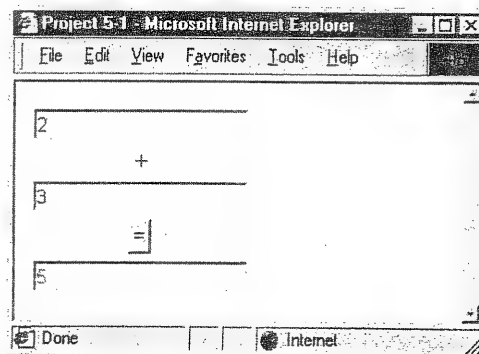
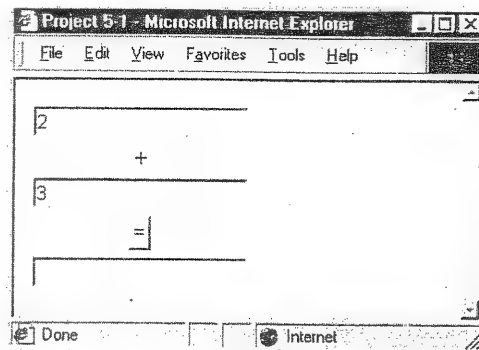
<TR>
<TD ALIGN="CENTER"><INPUT TYPE="SUBMIT" VALUE="="></TD>
</TR>

<TR>
<TD><INPUT TYPE="TEXT" NAME="REZULTAT"
VALUE ="<? echo $OP1 + $OP2 ?>"
disabled></TD>
</TR>

</TABLE>
</FORM>
</BODY>
</HTML>

```

2. Alocați puțin timp studiului scriptului PHP, acordând o atenție specială modului de utilizare a instrucțiunilor PHP echo pentru generarea valorilor atributelor HTML.
3. Orientați un browser Web spre adresa URL a fișierului încărcat în etapa anterioară. Ecranul browserului trebuie să fie asemănător celui prezentat în ilustrația următoare. Introduceți valorile celor doi operanzi și executați clic pe butonul de expediere, care este marcat cu un simbol al egalității.
4. În momentul executării scriptului de prelucrare, acesta obține accesul la valorile celor două variabile din formular care reprezintă operanzii, calculează suma lor și specifică suma ca fiind valoarea atributului VALUE a casetei cu text numită REZULTAT. Un rezultat caracteristic este prezentat în ilustrația următoare.



Proiect 5-2: Un calculator pentru date calendaristice

În cadrul acestui proiect, veți scrie și veți executa un program PHP care execută operații aritmetice cu date. Programul permite utilizatorului să introducă o dată și o distanță, exprimată în zile, în raport cu data respectivă. Apoi calculează și afișează data rezultantă.

Scopul proiectului

- Prezentarea modului în care datele calendaristice pot fi introduse sub formă de text, convertite în format numeric și utilizate în expresiile aritmetice

Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit p-5-2.php și încărcați acest fișier în serverul dumneavoastră PHP:

```

<HTML>
<HEAD>
<TITLE>Proiect 5-2</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-5-2.php -->
<FORM METHOD="POST" ACTION="p-5-2.php">
<TABLE>

<TR>
Data:<BR>
<TD><INPUT TYPE="TEXT" NAME="DATA" VALUE="<?php echo
$DATA ?>"></TD>
</TR>

<TR>
<TD ALIGN="CENTER">+</TD>
</TR>

<TR>
<TD><INPUT TYPE="TEXT" NAME="DISTANTA" VALUE="<?php echo
$DISTANTA ?>"></TD>
</TR>

<TR>
<TD ALIGN="CENTER"><INPUT TYPE="SUBMIT" VALUE="="></TD>
</TR>

<TR>
<TD><INPUT TYPE="TEXT" NAME="REZULTAT"
VALUE="<?php echo date("M j, Y",
(strtotime($DATA)+60*60*24*($DISTANTA))) ?>" disabled><TD>
</TR>

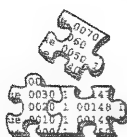
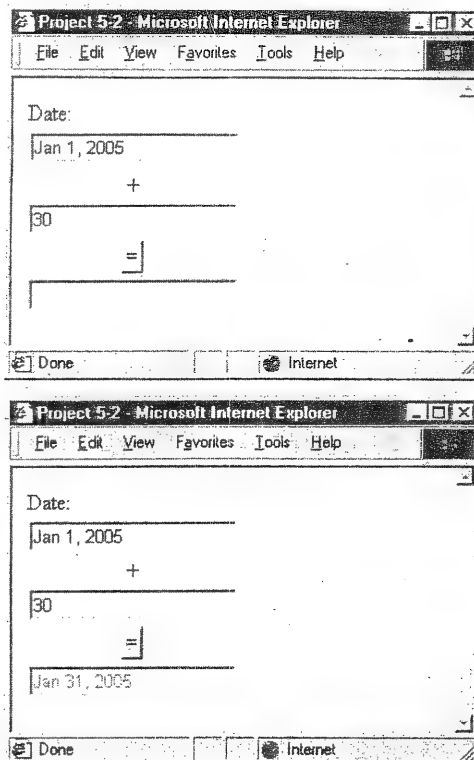
</TABLE>
</FORM>
</BODY>
</HTML>

```

2. Alocați un timp studiului scriptului PHP, acordând o atenție specială modului de calcul al valorii atributului VALUE al casetei cu text denumită REZULTAT. Sub-expresia 60*60*24 reprezintă numărul de secunde al unei zile. Funcția strtotime() preia o dată ca argument și returnează numărul secundelor scurse de la 1 ianuarie 1970, ora 00:00 GMT, moment cunoscut sub numele de *epocă UNIX*. Procedul de reprezentare a datelor și a orelor sub forma numărului de secunde scurse de la momentul epocii UNIX este cunoscut sub numele de *oră UNIX*. Funcția date() preia două argumente. Funcția formatează data furnizată de al doilea argument în conformitate cu șirul dat ca prim argument. Șirul

"M j, Y" determină funcția să returneze numele lunii, urmat de ziua din lună, o virgulă și valoarea anului.

3. Orientați un browser Web spre adresa URL a fișierului care conține scriptul pe care l-ați încărcat în cadrul primei etape. Ecranul browserului trebuie să fie asemănător celui prezentat în următoarea ilustrație. Introduceți o dată și o distanță (exprimată în zile) și executați clic pe butonul de expediere, care este etichetat cu un semn de egalitate.
4. La executarea scriptului de prelucrare, acesta obține accesul la valorile celor două variabile din formular, care reprezintă data și distanța exprimată în zile, calculează suma acestora și specifică suma ca valoare a atributului VALUE al casetei cu text denumită REZULTAT. Un rezultat caracteristic este prezentat în ilustrația următoare.



Proiect 5-3: Un generator de știri

În cadrul acestui proiect, veți scrie un program PHP care construiește un program de știri umoristice. Programul permite utilizatorului să scrie câteva știri de text și apoi assemblează știrile într-un articol.

Scopul proiectului

- Prezentarea modului de utilizare a știrilor text pentru a specifica un text HTML

Pas cu pas

1. Plasați următoarea pagină HTML într-un fișier denumit p-5-3.html și încărcăți acest fișier în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Proiect 5-3</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-5-3.html -->
<FORM METHOD="POST" ACTION="p-5-3.php">
<TABLE>
```

```
<TR>
<TD ALIGN="RIGHT">Prenumele utilizatorului:</TD>
<TD><INPUT TYPE="TEXT" NAME="USER"></TD>
</TR>

<TR>
<TD ALIGN="RIGHT">Oras mare:</TD>
<TD><INPUT TYPE="TEXT" NAME="ORAS"></TD>
</TR>

<TR>
<TD ALIGN="RIGHT">Inghetata preferata:</TD>
<TD><INPUT TYPE="TEXT" NAME="AROMA"></TD>
</TR>

<TR>
<TD ALIGN="RIGHT">Numele interpretului pop-rock preferat:</TD>
<TD><INPUT TYPE="TEXT" NAME="STAR"></TD>
</TR>

<TR>
<TD ALIGN="RIGHT">Numele unui rau cunoscut:</TD>
<TD><INPUT TYPE="TEXT" NAME="RAU"></TD>
</TR>

<TR>
<TD></TD>
<TD><INPUT TYPE="SUBMIT"></TD>
</TR>

</TABLE>
</FORM>
</BODY>
</HTML>
```

2. Plasați următorul script PHP într-un fișier denumit p-5-3.php și încărcăți acest fișier în serverul dumneavoastră PHP, plasându-l în același catalog ca și fișierul p-5-3.html:

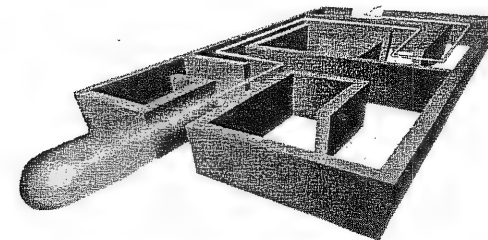
```
<HTML>
<HEAD>
<TITLE>Proiect 5-3</TITLE>
</HEAD>
<BODY>
<!-- Fișierul p-5-3.php -->
<CENTER><H1>Bursa stirilor despre celebritati</H1></CENTER>
<CENTER><?php echo date("l, F j, Y") ?></CENTER>
<CENTER><H2>ULTIMELE STIRI!!!</H2></CENTER>
<HR>
<?php echo $ORAS ?>:
<P>
Seara trecută, starul pop <?php echo $STAR ?> a fost vazut intr-o companie necunoscuta. Cei doi serveau inghetata de <?php echo $AROMA ?> la <?php echo $RAU ?> Club, un local cunoscut din <?php echo $ORAS ?> frecventat de personalitati. Conform unor surse confidentiale, misterioasa companie era <?php echo $USER ?>, fostul sofer al castigatorului de anul trecut al Premiului Grammy...
</BODY>
</HTML>
```


3. Studiați formularul HTML și identificați variabilele de tip formular pe care le definește. Apoi, studiați scriptul PHP și identificați locurile unde valorile variabilelor din formular sunt folosite pentru generarea de text HTML.
4. Orientați un browser Web spre adresa URL a fișierului care conține formularul HTML. Introduceți o valoare pentru fiecare câmp. Ecranul browserului trebuie să fie asemănător celui prezentat în următoarea ilustrație; dar, desigur, valorile dumneavoastră vor fi diferite de cele prezentate în figură. Executați clic pe butonul de expediere.
5. La executarea scriptului de prelucrare, acesta obține acces la valorile variabilelor din formular și le inserează sub formă de text HTML trimis browserului. Un rezultat caracteristic este prezentat în ilustrația următoare.

✓ Test de evaluare

1. Scrieți o instrucțiune care definește o constantă denumită VITEZA, care are valoarea 186,282*.
2. Scrieți o instrucțiune care afișează o valoare ce indică dacă a fost sau nu definită constanta LUNGIME.
3. Dacă variabila \$pisica are valoarea "Tom" și dacă variabila \$animal are valoarea "pisica", care este numele unei variabile dinamice cu valoarea "Tom"?
4. Dacă se procedează la împărțirea a două valori întregi, care este tipul rezultatului?
5. Dacă o valoare de tip întreg se împarte la o valoare de tip dublu, care este tipul rezultatului?
6. Scrieți o instrucțiune care modifică tipul variabilei \$cost în întreg.

* Este vorba despre viteza luminii, exprimată în mile pe secundă. – N. T.



Partea a II-a:

Scrierea unor programe PHP cu un grad avansat de complexitate

Modulul 6:

Scrierea instrucțiunilor PHP condiționale

Scopuri

- Învățați să definiți și să folosiți valorile de tip adevărat/fals
- Învățați să înțelegeți și să scrieți instrucțiunile `if` și instrucțiunile conexe
- Învățați să înțelegeți și să scrieți instrucțiunile `switch` și instrucțiunile conexe
- Învățați să înțelegeți și să scrieți instrucțiuni de ciclare, inclusiv instrucțiunile `while`, `do while` și `for`

Majoritatea programelor utile nu se comportă în exact același mod la fiecare rulare a acestora. În schimb, programele iau decizii, executând uneori o operație și alteori alta, în funcție de circumstanțe. De exemplu, un program util pentru calculul impozitului pe venit nu folosește aceeași rată a impozitului pentru fiecare contribuabil. În cadrul acestui modul, veți învăța să încorporați instrucțiunile condiționale în programele dumneavoastră, astfel încât programele să poată lua decizii.

Utilizarea valorilor de tip adevărat/fals

Programele PHP iau decizii prin evaluarea unor *expresii condiționale* și execută instrucțiuni bazate pe rezultatele acestor evaluări. Expresiile condiționale sunt asimilate ca având una din două valori: *true* (adevărat) sau *false* (fals). Uneori, expresiile condiționale se mai numesc și expresii booleene, în onoarea matematicianului care le-a studiat, George Boole. Constanta *true* are valoarea 1, iar constanta *false* are valoarea 0.

Puteți forma o expresie condițională folosind constanta *true* sau constanta *false*. O modalitate mai utilă de a forma o expresie condițională constă în utilizarea unui *operator relațional* pentru compararea a două valori numerice. Să luăm în considerare următorul exemplu:

```
$a < 1
```

Această expresie condițională folosește operatorul *<*, care are, în esență, aceeași semnificație ca în algebră. Expresia are valoarea *true* dacă și numai dacă valoarea variabilei *\$a* este mai mică decât unitatea; în toate celelalte cazuri, are valoarea *false*. PHP furnizează un set de asemenea operatori relaționali, prezentați pe scurt în tabelul 6-1.

PHP nu vă obligă să comparați numai valori numerice. Puteți folosi operatorii relaționali pentru compararea șirurilor; cu toate acestea, un șir care apare într-o expresie condițională este convertit la o valoare numerică înainte de evaluarea expresiei. Deseori, se ajunge la rezultate neașteptate. În general, valorile șirurilor trebuie să fie comparate folosind o funcție de comparare a șirurilor; aceste funcții vor fi explicate în Modulul 7.

Tabelul 6-1 Operatori relaționali ai limbajului PHP

Operator	Descriere
<i>a < b</i>	Adevărat dacă valoarea lui <i>a</i> este mai mică decât valoarea lui <i>b</i> .
<i>a > b</i>	Adevărat dacă valoarea lui <i>a</i> este mai mare decât valoarea lui <i>b</i> .
<i>a <= b</i>	Adevărat dacă valoarea lui <i>a</i> este mai mică sau egală cu valoarea lui <i>b</i> .
<i>a >= b</i>	Adevărat dacă valoarea lui <i>a</i> este mai mare sau egală cu valoarea lui <i>b</i> .
<i>a == b</i>	Adevărat dacă valoarea lui <i>a</i> este egală cu valoarea lui <i>b</i> .
<i>a != b</i>	Adevărat dacă valoarea lui <i>a</i> este diferită de valoarea lui <i>b</i> .
<i>a === b</i>	Adevărat dacă <i>a</i> și <i>b</i> sunt identice; cu alte cuvinte, dacă <i>a</i> și <i>b</i> au același tip și dacă valoarea lui <i>a</i> este egală cu valoarea lui <i>b</i> .
<i>a !== b</i>	Adevărat dacă <i>a</i> și <i>b</i> nu sunt identice; cu alte cuvinte, dacă <i>a</i> și <i>b</i> nu sunt de același tip sau dacă valoarea lui <i>a</i> este diferită de valoarea lui <i>b</i> .

Pentru comoditate, puteți forma expresii condiționale fără un operator relațional. De exemplu, dacă *\$a* este o variabilă numerică, puteți folosi expresia *\$a* ca expresie condițională. Expresia este considerată ca având valoarea *false* dacă valoarea variabilei *\$a* este zero, respectiv valoarea *true* dacă valoarea variabilei respective este

diferită de zero. Dacă folosiți un șir ca expresie condițională, expresia are valoarea *false* dacă șirul este vid sau dacă are valoarea specială *"\0"*, care simbolizează un octet cu valoarea zero. Similar, utilizarea unei valori nedefinite ca expresie condițională determină obținerea valorii *false*. Dacă folosiți un tablou sau un obiect ca expresie condițională, aceasta are valoarea *false* dacă tabloul sau obiectul sunt vide; în caz contrar, expresia are valoarea *true*.

Pentru a sintetiza, iată regulile care definesc rezultatul unei expresii condiționale:

- Constantele *true* și *false* își iau respectiv valorile lor booleene corespunzătoare.
- O expresie condițională care constă dintr-o valoare nedefinită are valoarea *false*; în caz contrar, rezultatul depinde de tipul valorii, în speță numeric, șir, tablou sau obiect.
- O expresie condițională care constă dintr-o valoare numerică are valoarea *false* dacă valoarea este zero; în caz contrar, are valoarea *true*.
- O expresie condițională care constă dintr-o valoare de tip șir are valoarea *false* dacă șirul este vid; în caz contrar, are valoarea *true*.
- O expresie condițională care constă dintr-o valoare de tip tablou sau obiect are valoarea *false* dacă tabloul sau obiectul sunt vide; în caz contrar, are valoarea *true*.
- O expresie condițională alcătuită dintr-un operator relațional și din operandii săi ia valori în conformitate cu rezultatul comparației (numerice sau nu).

Puteți forma expresii condiționale complexe prin unirea a două expresii condiționale cu ajutorul unui *operator logic*. De exemplu, expresia următoare este adevărată dacă ambele expresii condiționale care o compun sunt adevărate:

```
$a < 1 AND $b < 1
```

Cu alte cuvinte, expresia este adevărată dacă atât variabila *\$a*, cât și variabila *\$b* au valori mai mici decât unitatea. Tabelul 6-2 prezintă pe scurt operatorii logici ai limbajului PHP. Rețineți că puteți prefixa o expresie condițională cu operatorul *!*, care inversează valoarea „adevărat” sau „fals” a operandului său.

În general, expresiile sunt evaluate de la stânga la dreapta. Totuși, operatorii care apar în partea superioară a tabelului dispun de o precedență mai ridicată și sunt efectuați anterior operatorilor cu o precedență mai redusă, dacă nu sunt folosite paranteze pentru a specifica o altă ordine a operațiilor.

Tabelul 6-2 Operatori logici ai limbajului PHP

Operator	Descriere
<i>x AND y</i>	Adevărat dacă atât <i>x</i> , cât și <i>y</i> sunt adevărate.
<i>x && y</i>	Adevărat dacă atât <i>x</i> , cât și <i>y</i> sunt adevărate.
<i>x OR y</i>	Adevărat dacă minimum una din expresiile <i>x</i> și <i>y</i> este adevărată.
<i>x y</i>	Adevărat dacă minimum una din expresiile <i>x</i> și <i>y</i> este adevărată.
<i>x XOR y</i>	Adevărat dacă numai una din expresiile <i>x</i> și <i>y</i> este adevărată.
<i>!x</i>	Adevărat dacă <i>x</i> este falsă.

? Sfatul specialistului

Întrebare: De ce folosește PHP doi operatori logici reprezentând conjuncții (AND și &&) și doi operatori logici reprezentând disjuncții (OR și ||)? Care sunt diferențele între cele două tipuri de operatori?

Răspuns: Ambii operatori de conjuncție execută aceeași operație, ca de altfel și ambii operatori de disjuncție. Cu toate acestea, operatorii diferă din punctul de vedere al *precedenței* – caracteristica determinantă a ordinii în care sunt executate operațiile în timpul evaluării expresiilor. Operatorii && și || au o precedență relativ ridicată, în timp ce operatorii AND și OR au o precedență relativ redusă. Tabelul 6-3 indică precedența operatorilor PHP, conținând numeroși operatori care nu au fost încă prezentați.

Tabelul 6-3: Precedența operatorilor PHP

Operator

! ~ ++ -- (int) (double) (string) (array) (object)

* / %

+ -

< <= > >=

== != === !==

&

^

|

&&

||

?:

= += -= *= /= .= %= &= |= ^= -- <<= >>=

And

Xor

Or

De exemplu, să considerăm următoarea expresie:

```
$a + $b * $c
```

Precedența operatorului de multiplicare * este mai ridicată decât aceea a operatorului de adunare +, deci înmulțirea este efectuată prima, chiar dacă adunarea apare la stânga înmulțirii. Cu alte cuvinte, expresia este evaluată ca și cum ar fi fost scrisă astfel:

```
$a + ($b * $c)
```

Dacă doriți ca adunarea să fie efectuată prima, puteți folosi paranteze în cadrul expresiei, astfel:

```
($a + $b) * $c
```



Test „la minut”

Să presupunem că variabila \$a are valoarea 10, variabila \$b are valoarea 1, iar variabila \$c are valoarea 1:

- Care este valoarea expresiei \$a < \$b?
- Care este valoarea expresiei \$a <= \$b?
- Care este valoarea expresiei \$a > \$b?
- Care este valoarea expresiei \$b >= \$c?
- Care este valoarea expresiei \$a > \$b AND \$b >= \$c?

Scrierea instrucțiunilor if simple

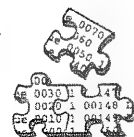
Expresiile condiționale nu sunt deosebit de interesante sau utile ca atare. Aceste expresii sunt însă esențiale pentru scrierea *instrucțiunilor condiționale*, prin care se iau decizii. Cea mai simplă instrucțiune condițională este instrucțiunea if, care execută două operații. Mai întâi, evaluează o expresie condițională. Apoi, dacă și numai dacă valoarea expresiei condiționale este true, instrucțiunea if execută o instrucțiune specificată.

Iată o instrucțiune if simplă:

```
if ($numar > 10)
    echo "Acesta este un numar mare";
```

Să ne reamintim că, în general, limbajul PHP ignoră spațiile albe. În mod convențional, o instrucțiune asociată unei instrucțiuni if este scrisă decalat în raport cu aceasta. Acest procedeu este recomandat deoarece prin utilizarea sa este facilitată citirea programului. Atunci când este executată, instrucțiunea if evaluează expresia condițională \$numar > 10, care este adevărată numai dacă valoarea variabilei \$numar este mai mare decât 10. Instrucțiunea echo este executată numai dacă valoarea variabilei \$numar este mai mare decât 10.

Pentru a acumula experiență în utilizarea instrucțiunii if, nu uitați să parcurgeți Proiectul 6-1.



Proiect 6-1: Testarea valorilor numerice

În cadrul acestui proiect, veți scrie și veți executa un mic program PHP, care include o instrucțiune if. Programul indică dacă un număr pe care îl introduceți este sau nu mai mare decât 10.

Răspunsuri la test:

- false
- false
- true
- true
- true

Scopul proiectului

- Prezentarea modului de funcționare a instrucțiunii `if`

Pas cu pas

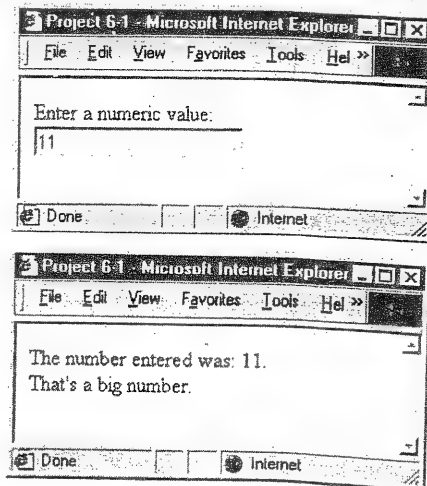
1. Plasați următorul script PHP într-un fișier denumit `p-6-1.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Proiect 6-1</TITLE>
</HEAD>
<BODY>
<!-- Fișier p-6-1.php -->
<?php
    echo "Numarul introdus a fost: $numar.";
    if ($numar > 10)
        echo "<BR>Acesta este un numar mare.";
?>
</BODY>
</HTML>
```

2. Plasați următoarea pagină HTML într-un fișier denumit `p-6-1.html` și încărcați acest fișier în serverul dumneavoastră, plasându-l în același catalog ca și fișierul `p-6-1.php`:

```
<HTML>
<HEAD>
<TITLE>Proiect 6-1</TITLE>
</HEAD>
<BODY>
<!-- Fișier p-6-1.html -->
<FORM METHOD="POST" ACTION="p-6-1.php">
    Introduceți o valoare numerică:
    <BR><INPUT TYPE="TEXT" NAME="numar">
</FORM>
</BODY>
</HTML>
```

3. Orientați un browser Web spre adresa URL a fișierului încărcat în etapa precedentă. Ecranul browserului trebuie să fie asemănător celui prezentat în ilustrația următoare. Introduceți un număr și apăsați pe tasta ENTER.*
4. În momentul execuției scriptului de prelucrare, acesta compară valoarea pe care ați introdus-o cu valoarea 10. Dacă ați introdus o valoare mai mare decât 10, scriptul afișează un mesaj. Un rezultat caracteristic este prezentat în ilustrația următoare.



* În original este menționată tasta RETURN, nume care nu se mai utilizează de mult în tastaturile calculatoarelor. – N.T.



Test „la minut”

- Scrieți o instrucțiune care afișează mesajul „aoleu!” dacă valoarea variabilei `$erori` este mai mare decât zero.
- Scrieți o instrucțiune `if` care afișează mesajul „nu trece” dacă valoarea variabilei `$culoare` este egală cu valoarea variabilei `$rosu` sau cu valoarea variabilei `$galben`.

Scrierea unor instrucțiuni `if` mai complexe

Să presupunem că doriți să executați nu una, ci două instrucțiuni în cazul în care o anumită expresie condițională este adevărată. Puteți scrie două instrucțiuni `if`, câte una pentru fiecare dintre instrucțiunile pe care doriți să le executați. Alternativ, puteți crea un *grup de instrucțiuni*, prin includerea unei serii de instrucțiuni între paranteze accolade. Un grup de instrucțiuni se comportă ca o singură instrucțiune și se poate asocia cu o instrucțiune `if`. Să considerăm următorul exemplu:

```
if ($numar > 10)
{
    echo "<BR>Numarul este mai mare decat 10.";
    echo "<BR>Deci, trebuie sa fie mare.";
}
```

În cazul în care expresia condițională are valoarea `true`, sunt executate ambele instrucțiuni din cadrul grupului de instrucțiuni. Unii programatori preferă să formateze programe ca acesta în alt mod. De exemplu, ei pot scrie ceva de genul următor:

```
if ($numar > 10) {
    echo "<BR>Numarul este mai mare decat 10.";
    echo "<BR>Deci, trebuie sa fie mare.";
}
```

Acest stil este mai compact, dar face dificilă identificarea parantezei acolade de deschidere corespunzătoare parantezei de închidere. Probleme de lizibilitate de acest gen devin importante la scrierea unor instrucțiuni `if` mai complicate.

Să presupunem că doriți să executați o instrucțiune atunci când o condiție este adevărată și o altă instrucțiune când condiția este falsă. Instrucțiunea `else` vă permite să procedați astfel. Să luăm în considerare următorul exemplu:

```
if ($numar > 10)
    echo "<BR>Acesta este un numar mare.";
else
    echo "<BR>Acesta este un numar mic.";
```

Răspunsuri la test:

- `if ($erori > 0)`
 `echo "aoleu!";`
- `if ($culoare == $rosu OR $culoare == $galben)`
 `echo "nu trece";`

În acest exemplu, mesajul „Acesta este un număr mare” este afișat atunci când valoarea variabilei \$numar este mai mare decât 10; mesajul „Acesta este un număr mic” este afișat în caz contrar. Dacă doriți, puteți folosi o instrucțiune else cu un grup de instrucțiuni. De exemplu:

```
if ($numar > 10)
    echo "<BR>Acesta este un numar mare.";
else
{
    echo "<BR>Numarul este mai mic decat 10.";
    echo "<BR>Este un numar mic.";
}
```

Instrucțiunea asociată unei instrucțiuni if sau else poate fi ea însăși o instrucțiune if. O asemenea instrucțiune if se numește *instrucțiune if imbricată*. Iată un exemplu de instrucțiune if imbricată:

```
if ($numar > 10)
    if ($numar > 100)
        echo "<BR>Acesta este un numar foarte mare.";
    else
        echo "<BR>Acesta este un numar mare.";
else
    echo "<BR>Acesta este un numar mic.";
```

Exemplul afișează mesajul „Acesta este un număr foarte mare.” dacă valoarea variabilei \$numar depășește 100; în caz contrar, dacă valoarea variabilei \$numar este mai mare decât 10, se afișează mesajul „Acesta este un număr mare.”. Dacă valoarea variabilei \$numar este mai mică sau egală cu 10, exemplul afișează mesajul „Acesta este un număr mic.”.

Instrucțiunile if imbricate pot deveni extrem de dificil de înțeles dacă numărul de instrucțiuni și nivelul de imbricare nu sunt relativ reduse. Deci trebuie să le folosiți cu economie.

O instrucțiune corelată atât cu instrucțiunea if, cât și cu instrucțiunea else, este instrucțiunea elseif. Când este folosită corect, poate fi mai simplu de înțeles decât o instrucțiune if imbricată, logic echivalentă cu aceasta. Iată un exemplu de instrucțiune elseif:

```
if ($numar > 100)
    echo "<BR>Acesta este un numar foarte mare.";
elseif ($numar > 10)
    echo "<BR>Acesta este un numar mare.";
elseif ($numar > 1)
    echo "<BR>Acesta este un numar mic.";
else
    echo "<BR>Acesta este un numar foarte mic.";
```

Exemplul extinde funcționalitatea exemplului anterior, afișând mesajul „Acesta este un număr foarte mic.” pentru valori ale variabilei \$numar mai mici sau egale cu 1. Într-un caz general, cu o instrucțiune if și cu o instrucțiune else poate fi asociat

un număr mult mai mare de instrucțiuni elseif. PHP evaluează expresiile condiționale în mod succesiv, pornind de la expresia condițională asociată instrucțiunii if. PHP execută instrucțiunea asociată primei expresii condiționale care are valoarea true; dacă nici o expresie condițională nu are valoarea true, PHP execută instrucțiunea asociată cu instrucțiunea else. Este permisă omiterea instrucțiunii else, caz în care nu este executată nici o instrucțiune dacă nici una din expresiile condiționale nu are valoarea true.



Test „la minut”

- Scrieți o instrucțiune if și o instrucțiune else pentru a afișa mesajul „stai” dacă valoarea variabilei \$culoare este egală cu valoarea variabilei \$rosu sau cu valoarea variabilei \$galben, respectiv pentru a afișa mesajul „liber” în caz contrar.
- Scrieți o instrucțiune if și o instrucțiune else pentru a înmulți valoarea variabilei \$numar cu 10 dacă valoarea variabilei \$factor este 1, respectiv pentru a înmulți valoarea variabilei \$numar cu 100 în caz contrar.

Scrierea instrucțiunilor switch, break și default

Instrucțiunea if vă permite să luați o decizie în două sensuri. Pentru a putea lua o decizie în mai multe sensuri, puteți folosi mai multe instrucțiuni if, else sau elseif. Cu toate acestea, când doriți ca programul dumneavoastră să aleagă dintr-un set de alternative care pot fi reprezentate prin valori întregi, instrucțiunea switch este o opțiune mai convenabilă.

De exemplu, să presupunem că valoarea variabilei \$numar este 1, 2 sau 3, reprezentând respectiv dimensiunile mică, medie și mare. Iată un mic program care afișează dimensiunile asociate valorilor variabilei \$numar:

```
switch($numar)
{
    case(1):
        echo "mic";
        break;
```

Răspunsuri la test:

- if (\$culoare == \$rosu OR \$culoare == \$galben)


```
    echo "stai";
        else
            echo "liber";
```
- if (\$factor == 1)


```
    $numar = 10 * $numar;
        else
            $numar = 100 * $numar;
```



```

case(2):
    echo "mediu";
    break;
case(3):
    echo "mare";
    break;
default:
    echo "Acesta nu este un cod valabil";
}

```

Ațiunea unei instrucțiuni switch este determinată de valoarea unei expresii întregi, nu de valoarea unei expresii condiționale. Numele variabilei este dat între parantezele care urmează după cuvântul cheie switch. Parantezele accolade delimitează o serie de instrucțiuni case și o instrucțiune default opțională, fiecare dintre instrucțiunile cuprinse între paranteze putând avea instrucțiuni asociate. Când este executată, instrucțiunea switch încearcă să stabilească o identitate între valoarea variabilei sale asociate și valoarea asociată unei instrucțiuni case. Se vor executa instrucțiunile asociate primei instrucțiuni case pentru care identitatea respectivă este valabilă. Dacă valoarea variabilei din instrucțiunea switch nu corespunde nici uneia din valorile asociate instrucțiunilor case, se vor executa instrucțiunile asociate instrucțiunii default, dacă există o asemenea instrucțiune.

Un procedeu de programare indicat constă în aceea ca fiecare instrucțiune case din cadrul unei instrucțiuni switch să se încheie cu o instrucțiune break. Instrucțiunea break determină încheierea execuției instrucțiunii switch, transferând controlul următoarei instrucțiuni secvențiale care succede instrucțiunii switch. În absența instrucțiunii break, execuția trece la următoarea instrucțiune case sau default, fapt nedorit în majoritatea cazurilor.

Sugestie

Nu este necesar să folosiți numere întregi consecutive în instrucțiunile case ale unei instrucțiuni switch. Dacă preferați, puteți folosi numere întregi non-consecutive, numere cu virgulă mobilă sau șiruri.

? Sfatul specialistului

Întrebare: PHP include numeroase instrucțiuni condiționale, incluzând instrucțiunile if, else, elseif și switch. Există și alte mecanisme de luare a deciziilor?

Răspuns: Da. Operatorul condițional `?:`, denumit uneori operator ternar sau operator întrebare-două puncte, constituie o altă modalitate de a scrie decizii în PHP. Operatorul condițional formează o expresie care se poate folosi în multe contexte PHP. Iată sintaxa de utilizare a acestuia:

expresie-condiționala ? valoare-adevarat : valoare-fals

Observați cum semnul întrebării este separat de caracterul două puncte prin valoarea valoare-adevarat. Operatorul condițional își evaluează expresia condițională. Dacă expresia este evaluată la valoarea true (adevărat), operatorul condițional returnează valoarea valoare-adevarat; în caz contrar, returnează valoarea valoare-fals. Operatorul condițional vă permite să specificați deciziile într-o manieră foarte concisă.

De exemplu, să luăm în considerare următoarea instrucțiune de atribuire, care folosește un operator condițional:

```
$a = ($b > $c) ? 1 : 2
```

Această instrucțiune de atribuire compară valorile variabilelor \$b și \$c. Dacă valoarea variabilei \$b este mai mare decât aceea a variabilei \$c, atunci variabilei \$a îi este atribuită valoarea 1; în caz contrar, variabilei respective îi este atribuită valoarea 2.



Test „la minut”

- Scrieți o instrucțiune switch care testează valoarea variabilei \$exponent. Instrucțiunea trebuie să atribuie variabilei \$factor o valoare după cum urmează: dacă \$exponent este 1, \$factor primește valoarea 10; dacă \$exponent este 2, \$factor primește valoarea 100; altfel, \$factor primește valoarea 0.
- În programare, se recomandă ca fiecare instrucțiune case să fie asociată cu o instrucțiune _____.

Scrierea instrucțiunilor for

Instrucțiunea for este o instrucțiune buclă sau o instrucțiune iterativă; cu alte cuvinte, o instrucțiune care execută în mod repetat instrucțiunile asociate. Iată un exemplu de utilizare a unei instrucțiuni for:

Răspunsuri la test:

```

• switch($exponent)
{
    case 1:
        $factor = 10;
        break;
    case 2:
        $factor = 100;
        break;
    default:
        $factor = 0;
}
• break

```

```
$suma = 0;
for ($n = 1; $n <= 3; $n++)
    $suma += $n;
echo "<BR>Suma intregilor de la 1 la $numar este $suma.";
```

În exemplu se calculează suma întregilor cuprinși între 1 și 3. Pentru aceasta, mai întâi se inițializează variabila \$suma la valoarea 0. Apoi, se execută o instrucțiune for care incrementează în mod repetat valoarea variabilei \$suma.

Pentru a vedea cum funcționează mecanismul acestei instrucțiuni, să examinăm componentele instrucțiunilor for. Instrucțiunea for include trei expresii, care apar între paranteze; fiecare expresie este separată de vecina sa printr-un caracter punct și virgulă. De asemenea, instrucțiunea for include o instrucțiune sau un grup de instrucțiuni, cunoscute sub numele de *corpul* instrucțiunii for. În exemplul de mai sus, instrucțiunea \$suma += \$n este corpul instrucțiunii for.

Să examinăm mai amănunțit cele trei expresii:

- Prima expresie este expresia de *inițializare*. Aceasta se execută atunci când PHP ajunge la instrucțiunea `for`. În exemplu, expresia de inițializare atribuie valoarea 1 variabilei `$n`, variabilă denumită *variabilă de ciclare* sau *index*.
- A doua expresie este *expresia de test*. Aceasta este o expresie condițională care indică dacă se execută sau nu corpul instrucțiunii. În general, face referire la variabila de ciclare. În cadrul exemplului, expresia de test compară valoarea variabilei `$n` cu valoarea 3. Expresia de test este evaluată pentru prima dată imediat după evaluarea expresiei de inițializare.
- Cea de-a treia expresie este *expresia pas*. În general, aceasta modifică una sau mai multe variabile la care se face referire în expresia test. În cadrul exemplului, expresia `pas` incrementează valoarea variabilei `$n`.

Secvența de execuție a unei instrucțiuni for este următoarea:

1. Se evaluează expresia de inițializare.
2. Se evaluează expresia test.
3. Dacă rezultatul evaluării expresiei test este false, se execută etapa 7.
4. Se execută corpul buclei.
5. Se evaluează expresia pas.
6. Se trece la etapa 2.
7. Se încheie execuția instrucțiunii for, prin executarea următoarei instrucțiuni secvențiale.

Instrucțiunea for este utilă pentru numărare și executarea în mod repetat a unor acțiuni. Ca alt exemplu, iată o instrucțiune for care creează numeroase controale de tip buton. Numărul controalelor create este determinat de valoarea variabilei \$numar:

```
for($n=0; $n<=$numar; $n++)
    echo "<BR><INPUT TYPE=\"BUTTON\" VALUE=\"$n\">\n";
```

Așa cum veți vedea în Modulul 8, instrucțiunea `for` este utilă mai ales în lucrul cu tablouri.



Test „la minut“

- Scrieți o instrucțiune for care calculează suma întregilor cuprinși între 1 și 100.
- Scrieți o instrucțiune for care afișează la ieșire etichete HTML `
`. Numărul etichetelor afișate trebuie să fie egal cu valoarea variabilei \$n\$.

Scrierea instrucțiunilor while și do while

Practic, instrucțiunile `while` și `do while` reprezintă versiuni „manuale“ ale instrucțiunii `for`. Dacă o instrucțiune `for` are trei expresii, o instrucțiune `while` sau `do while` are una singură, și anume expresia de test. Așa cum se întâmplă de obicei, expresiile unei instrucțiuni `for` sunt opționale; fără o expresie de inițializare sau o expresie pas, o instrucțiune `for` operează în același mod ca o instrucțiune `while`. În consecință, următoarele două instrucțiuni sunt echivalente:

```
for ( ; $i <= 3; ) $suma += $i;
while ( $i <= 3 ) $suma += $i;
```

Când folosiți o instrucțiune `while`, trebuie să furnizați un mecanism oarecare, analog expresiei de incrementare a instrucțiunii `for`, care actualizează variabilele la care se face referire în expresia de test. De asemenea, sunteți responsabil cu inițializarea tuturor valorilor folosite în expresia de test.

Secvența de execuție a unei instrucțiuni while este următoarea:

1. Se evaluează expresia test.
2. Dacă rezultatul este false, se trece la etapa 5.
3. Se execută corpul buclei.
4. Se trece la etapa 1.
5. Se părăsește bucla, prin executarea următoarei instrucțiuni secvențiale.

Iată un exemplu care utilizează instrucțiunea `while`:

```
$n = 0;
$suma = 0;
while ($n <= 3)
{
```

Răspunsuri la test:

- ```

• $suma = 0;
 for ($i = 1; $i <= 100; $i++)
 $suma = $suma + $i;
• for ($i = 1; $i <= $n; $i++)
 echo "
";

```

```
$suma = $suma + $n;
$n++;
}
echo "Suma este $suma."
```

Remarcați că instrucțiunea `$n = 0;` inițializează bucla și că instrucțiunea `n++;` incrementează valoarea variabilei buclă `$n`. Instrucțiunea `while` este cel mai utilă atunci când un alt program necesar execută deja aceste funcții; în asemenea situații, instrucțiunea `while` este mai clară decât o instrucțiune `for` degenerată, căreia îi lipsesc una sau mai multe dintre expresiile sale obișnuite.

Instrucțiunea `do while` este oarecum asemănătoare instrucțiunii `while`. Diferența este aceea că instrucțiunea `do while` își execută corpul *înainte* de a-și evalua expresia de test. Astfel, corpul buclei unei instrucțiuni `do while` este întotdeauna executat cel puțin o dată; corpul unei instrucțiuni `while` este omis dacă expresia de test are inițial valoarea `false`.

Secvența de execuție a unei instrucțiuni `do while` este următoarea:

1. Se execută corpul buclei.
2. Se evaluează expresia de test.
3. Dacă rezultatul este adevărat, se trece la etapa 1.
4. Se încheie execuția buclei, prin executarea următoarei instrucțiuni secvențiale.

Iată un exemplu care folosește o instrucțiune `do while`. Observați că amplasarea expresiei de test imediat după corpul buclei vă reamintește faptul că executarea corpului are loc înainte de evaluarea expresiei de test:

```
$suma = 0;
$n = 1;
do
{
 $suma += $n;
 $n++;
}
while ($n <= $numar);
echo "
Suma întregilor cuprinși între 1 și $numar este $suma.";
```



### Sfatul specialistului

**Întrebare:** Instrucțiunea `while` și instrucțiunea `do while` par foarte asemănătoare. Când trebuie să folosesc o instrucțiune `do while` în locul unei instrucțiuni `while`?

**Răspuns:** Experții în programare au demonstrat că este posibilă scrierea oricărui program fără a folosi nici o instrucțiune `do while`. Deci, utilizarea instrucțiunii `do while` este o chestiune de comoditate, nu de necesitate. Vă puteți descurca folosind numai instrucțiuni `while`.

În general, este bine să fiți prudent și să folosiți instrucțiunea `while`, care evaluează o expresie de test înainte de executarea corpului acesteia. Astfel, se evită executarea eronată a corpului buclei. Totuși, când vedeți un model ca acesta:

```
<o actiune oarecare>;
while (expresie-test)
{
 <aceeasi actiune>
}
```

atunci puteți înlocui liniștit programul respectiv cu un program care folosește o instrucțiune `do while`:

```
do
{
 <o actiune oarecare>;
} while (expresie-test);
```

### Test „la minut”



- Scrieți o instrucțiune `while` și instrucțiunile asociate care afișează suma întregilor cuprinși între 1 și 100.
- Scrieți o instrucțiune `do while` și instrucțiunile asociate care afișează suma întregilor cuprinși între 1 și 100.



### Proiect 6-2: Validarea datelor de intrare introduse de utilizator

În cadrul acestui proiect, veți crea un formular HTML și un script PHP care permit unui utilizator să introducă date personale de categoria celor folosite într-o agendă de adrese e-mail. Scriptul PHP validează datele introduse de utilizator, garantând existența datelor în câmpurile obligatorii.

#### Scopurile proiectului

- Prezentarea modului de utilizare a instrucțiunilor condiționale
- Prezentarea unui mod de validare a datelor dintr-un formular

#### Răspunsuri la test:

```
$suma = 0;
$n = 1;
while ($n <= 100)
 $suma += $n;
echo "suma este $suma.";
```

```
$suma = 0;
$n = 1;
do
{
 $suma += $n;
}while ($n <= 100)
echo "suma este $suma.";
```

## Pas cu pas

1. Plasăți următorul script PHP într-un fișier denumit p-6-2.php și încărcați acest fișier în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Proiect 6-2</TITLE>
</HEAD>
<BODY>
<!-- Fisierul p-6-2.php -->
<?php

$erori=0;
if(!trim($porecla))
{
 echo "
Porecla este obligatorie.";
 $erori++;
}

if(!trim($prenume))
{
 echo "
Prenamele este obligatoriu.";
 $erori++;
}

if(!trim($nume))
{
 echo "
Numele este obligatoriu.";
 $erori++;
}

if(!trim($email))
{
 echo "
Adresa primara de e-mail "
 "este obligatorie.";
 $erori++;
}

if ($erori > 0)
 echo "

Va rugam folositi butonul Back "
 "al browserului dumneavoastra pentru a reveni la "
 "formular, corectati ";
if ($erori == 1)
 echo "eroarea, ";
if ($erori > 1)
 echo "erorile, ";
if ($erori > 0)
 echo "si reexpediatii formularul.";

?>
</BODY>
</HTML>
```

2. Plasăți următoarea pagină HTML într-un fișier denumit p-6-2.html și încărcați acest fișier în serverul dumneavoastră, plasând fișierul în același catalog cu fișierul p-6-2.php:

```
<HTML>
<HEAD>
<TITLE>Proiect 6-2</TITLE>
</HEAD>
<BODY>
<!-- Fisier p-6-2.html -->
<FORM METHOD="POST" ACTION="p-6-2.php">
<H1>Informatii privind persoana de contact</H1>
<TABLE>

<TR>
<TD>Porecla:</TD>
<TD><INPUT TYPE="TEXT" NAME="porecla"></TD>
</TR>

<TR>
<TD>Titlu:</TD>
<TD><INPUT TYPE="TEXT" NAME="titlu"></TD>
</TR>

<TR>
<TD>Prenume:</TD>
<TD><INPUT TYPE="TEXT" NAME="prenume"></TD>
</TR>

<TR>
<TD>Prenumele tatalui:</TD>
<TD><INPUT TYPE="TEXT" NAME="prenume_tata"></TD>
</TR>

<TR>
<TD>Nume:</TD>
<TD><INPUT TYPE="TEXT" NAME="Nume"></TD>
</TR>

<TR>
<TD>Adresa de e-mail principala:</TD>
<TD><INPUT TYPE="TEXT" NAME="email"></TD>
<TD WIDTH="20"> </TD>
<TD>Adresa de e-mail secundara:</TD>
<TD><INPUT TYPE="TEXT" NAME="emailsecundar"></TD>
</TR>

<TR>
<TD>Numele companiei:</TD>
<TD><INPUT TYPE="TEXT" NAME="nume_companie"></TD>
</TR>

<TR>
<TD>Adresa firmei:</TD>
<TD><INPUT TYPE="TEXT" NAME="adresa_firmei1"></TD>
<TD WIDTH="20"> </TD>
<TD>Adresa la domiciliu:</TD>
<TD><INPUT TYPE="TEXT" NAME="adresa_acasa"></TD>
</TR>
```

```

<TR>
 <TD></TD>
 <TD><INPUT TYPE="TEXT" NAME="adresa_firmei2"></TD>
</TR>

<TR>
 <TD>Oras:</TD>
 <TD><INPUT TYPE="TEXT" NAME="oras_birou"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD><INPUT TYPE="TEXT" NAME="oras_acasa"></TD>
</TR>

<TR>
 <TD>Stat:</TD>
 <TD><INPUT TYPE="TEXT" NAME="stat_birou"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD><INPUT TYPE="TEXT" NAME="stat_acasa"></TD>
</TR>

<TR>
 <TD>Cod postal:</TD>
 <TD><INPUT TYPE="TEXT" NAME="cod_birou"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD><INPUT TYPE="TEXT" NAME="cod_acasa"></TD>
</TR>

<TR>
 <TD>Telefon:</TD>
 <TD><INPUT TYPE="TEXT" NAME="telefon_birou"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD><INPUT TYPE="TEXT" NAME="telefon_acasa"></TD>
</TR>

<TR>
 <TD>Data nasterii:</TD>
 <TD><INPUT TYPE="TEXT" NAME="data_nastere"></TD>
</TR>

<TR>
 <TD>Numele sotului/sotiei:</TD>
 <TD><INPUT TYPE="TEXT" NAME="nume_sot"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD>Numele copiilor:</TD>
 <TD><INPUT TYPE="TEXT" NAME="copii"></TD>
</TR>

<TR>
 <TD>Ziua nuntii:</TD>
 <TD><INPUT TYPE="TEXT" NAME="zi_nunta"></TD>
</TR>
</TABLE>

```

```


<INPUT TYPE="SUBMIT" VALUE="Trimite">

<INPUT TYPE="RESET" VALUE="Sterge datele">

</FORM>
</BODY>
</HTML>

```

3. Dedicăți un interval de timp studiului scriptului PHP, acordând o atenție specială modului de utilizare a instrucțiunilor condiționale pentru validarea datelor din formular.
4. Orientați un browser Web spre adresa URL a fișierului HTML încărcat anterior. Ecranul browserului trebuie să fie asemănător celui prezentat în figura 6-1. Introduceți valori în mai multe câmpuri și apoi executați clic pe butonul „Trimite”.

The screenshot shows a web browser window titled "Project 6-2: Microsoft Internet Explorer". The browser's address bar is empty. The main content area displays a form titled "Contact Information". The form consists of several text input fields arranged in two columns. The left column contains fields for Nickname, Title, First Name, Middle Name, Last Name, Primary Email, Company Name, Office Address, City, State, Zip, Phone, Birthday, Spouse Name, and Anniversary. The right column contains fields for Secondary Email, Home Address, and Children's Names. At the bottom of the form, there are two buttons: "Submit" and "Clear the Form". The browser's status bar at the bottom shows "Done" and "Internet".

Figura 6-1 Formularul de introducere a datelor pentru proiectul 6-2.

5. La executarea scriptului, acesta verifică dacă porecla, prenumele, numele și adresa de e-mail există; dacă vreunul din aceste câmpuri lipsește, scriptul afișează un mesaj de eroare. Un rezultat caracteristic este prezentat în figura 6-2.

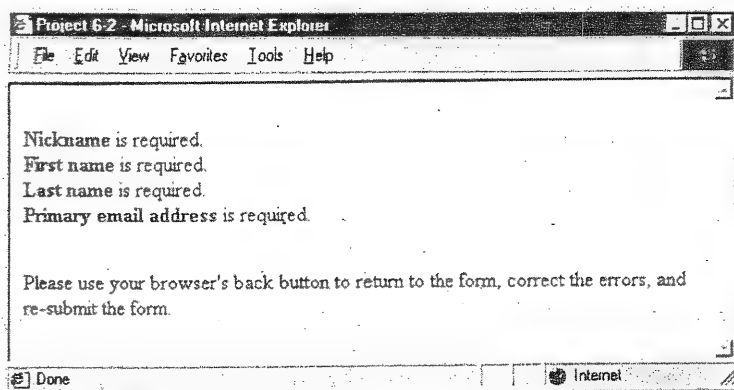


Figura 6-2 Datele de ieșire ale proiectului 6-2, afișând erorile de validare.



### Test de evaluare

1. Scrieți o instrucțiune `if` care atribuie variabilei `$y` valoarea 1 dacă variabila `$x` are valoarea 1, în caz contrar atribuind variabilei `$y` valoarea 2.
2. Scrieți o instrucțiune `switch` care atribuie valoarea 5 variabilei `$y` dacă variabila `$x` are valoarea 1, respectiv valoarea 15 dacă variabila `$x` are valoarea 2, valoarea 20 dacă variabila `$x` are valoarea 3, valoarea -1 în celelalte situații.
3. Scrieți o buclă `for` care are ca date de ieșire o serie de asteriscuri; numărul asteriscurilor trebuie să fie dat de valoarea variabilei `$stele`.
4. Scrieți o instrucțiune `if` care atribuie variabilei `$y` valoarea 1 dacă variabila `$x` are valoarea 1, respectiv valoarea 3 dacă variabila `$x` are valoarea 2, valoarea 5 dacă variabila `$x` are valoarea 3, valoarea -1 în celelalte situații.

## Modulul 7: Utilizarea funcțiilor

### Scopuri

- Învățați să apelați funcțiile
- Învățați să atribuiți unei variabile numele unei funcții
- Învățați să ascundeți mesajele de eroare generate în timpul execuției unei funcții
- Învățați să folosiți programele rezidente în fișierele externe
- Învățați să definiți o funcție
- Învățați să folosiți variabile globale, locale și statice
- Învățați să folosiți referințele

În Modulul 2 a fost prezentată noțiunea de funcție, care simplifică efectuarea unei varietăți de calcule. În cadrul acestui modul, veți învăța mai multe despre utilizarea funcțiilor și despre modul de definire a propriilor dumneavoastră funcții.

### Utilizarea unei funcții

Procesul de executare a unei funcții se numește *utilizarea, apelarea sau invocarea* funcției. Pentru a folosi o funcție, scrieți numele funcției, urmat de o pereche de paranteze. De exemplu, funcția `rand()`, care generează un număr întreg aleator, poate fi apelată astfel:

```
rand();
```

Majoritatea funcțiilor preiau *argumente*, reprezentând valori de intrare care influențează operarea și rezultatul funcției. Pentru a specifica argumente, acestea se înserază între paranteze; dacă specificați mai mult de un argument, fiecare argument trebuie separat de vecinul său prin intermediul unei virgule. Argumentul unei funcții poate fi o valoare literală, o variabilă sau o expresie.

Unele funcții PHP au argumente opționale, care pot fi specificate sau omise, în conformitate cu intențiile dumneavoastră. De exemplu, funcția `rand()` are două argumente opționale. Primul argument al funcției indică valoarea întreagă aleatoare cea mai mică pe care o va returna funcția; al doilea argument indică valoarea cea mai mare. Dacă omiteți ambele argumente, funcția returnează o valoare cuprinsă între 0 și cel mai mare rezultat posibil. Dacă doriți, puteți specifica primul argument, omițându-l pe al doilea:

```
rand(100);
```



Astfel, funcția este obligată să returneze o valoare cuprinsă între 100 și cel mai mare rezultat posibil.

Ca și `rand()`, majoritatea funcțiilor returnează valori. Puteți folosi într-o expresie valoarea returnată de o funcție. O situație foarte frecventă în care se procedează astfel o constituie utilizarea valorii returnate de o funcție într-o expresie de atribuire, astfel încât valoarea să fie accesibilă în mod repetat fără a se invoa funcția de mai multe ori. De exemplu:

```
$gogoasa = rand(1, 12);
```

Când se produce o eroare în timpul execuției unei funcții, PHP generează mesaje de eroare. Uneori, asemenea mesaje de eroare sunt nedorite. În acest caz, puteți suprima generarea mesajelor de eroare prin prefixarea numelui funcției invocate cu ajutorul caracterului `@`. De exemplu, pentru a suprima mesajele de eroare care pot apărea în timpul execuției funcției `f()`, invocați această funcție după cum urmează:

```
y = @f(x);
```



### Test „la minut”

- Scrieți o instrucțiune care atribuie variabilei `$x` un număr întreg aleator cuprins între 1 și 10.
- Scrieți o instrucțiune care invocă funcția `sensibil_la_eroari()`, care nu preia nici un argument și nu returnează nici un rezultat, suprimând toate mesajele de eroare rezultate.

## Utilizarea fișierelor incluse

Funcțiile PHP vă permit să obțineți accesul la programe PHP scrise anterior, create de dumneavoastră sau de către un alt programator în limbajul PHP. Un alt mecanism care vă permite să obțineți accesul la programele scrise anterior îl constituie instrucțiunea `require`, care are următoarea formă:

```
require(ume_fisier);
```

Argumentul `ume_fisier` are forma unui șir, deci o instrucțiune `require` caracteristică poate avea următorul aspect:

```
require("fisierul.inc");
```

Când este încărcat un script PHP care conține o instrucțiune `require`, conținutul fișierului specificat – cunoscut sub numele de *fișier de includere* – este inserat în script,

Răspunsuri la test:

- `$x = rand(1, 10);`
- `@sensibil_la_eroari();`

înlocuind instrucțiunea `require`. Dacă fișierul de includere conține linii de program PHP, trebuie să includă etichetele `<?php` și `?>`, amplasate în locațiile adecvate.

Deși se obișnuiește ca un fișier de includere să primească extensia de fișier `inc`, nu este obligatoriu să procedați astfel. Unii programatori PHP preferă să folosească extensia de fișier `php` pentru toate fișierele pe care le creează.

Instrucțiunea `require` vă poate scuti de un mare volum de muncă. De exemplu, să presupunem că scrieți o aplicație PHP care este alcătuită din mai multe scripturi, iar fiecare script afișează o pagină HTML care conține informații standard în partea de sus a paginii. Puteți crea un fișier script special, denumit `antet.inc`, care conține următoarele linii de program:

```
<HTML>
<HEAD>
<TITLE>Aplicatia care pune capac la toate aplicatiile</TITLE>
</HEAD>
<BODY>
<H1>Aceasta este aplicatia care incheie toate aplicatiile</H1>
<H5>Copyright 2005, Fane Programatorul si Compania SRL.</H5>
si asa mai departe...
```

Prin inserția instrucțiunii

```
require("antet.inc");
```

la începutul fiecărui script, determinați programul PHP să includă conținutul celui fișier ca și cum conținutul respectiv ar face parte din acel script. Acest procedeu poate simplifica întreținerea programului, deoarece informațiile standard pot fi rezidente într-un singur fișier, ceea ce le face ușor de localizat și revizuit.



### Sfatul expertului

**Întrebare:** Dacă doresc să obțin numele unui fișier de includere din program? Cum pot proceda?

**Răspuns:** Instrucțiunea `require` este prelucrată la încărcarea scriptului PHP asociat, înaintea legării valorilor la variabilele corespunzătoare. În consecință, nu puteți folosi o expresie pentru a specifica numele fișierului care urmează a fi inclus de către o instrucțiune `require`.

Și totuși puteți folosi instrucțiunea `include`, care este o instrucțiune executabilă ce determină evaluarea scriptului PHP dintr-un fișier specificat. De exemplu, instrucțiunea `include` din următorul program evaluează fișierul `fisier1`:

```
$x = 1;
include("fisier" . $x . ".inc");
```

La evaluarea fișierului de includere, instrucțiunile PHP pe care le conține sunt executate ca și cum ar fi apărut în textul scriptului apelant.

Instrucțiunea corelată `require_once` asigură faptul că fișierul specificat este inclus o singură dată într-un script dat. În cazul în care creați fișiere de includere care folosesc instrucțiunea `require` pentru a include conținutul altor fișiere de includere, puteți găsi instrucțiunea `require_once` utilă.



### Test „la minut”

- Scrieți o instrucțiune care include conținutul fișierului `subsol.inc` în textul sursă curent.
- Scrieți o instrucțiune care include conținutul fișierului `subsol.inc` în textul sursă curent, asigurându-vă că fișierul este inclus o singură dată.

## Definirea unei funcții

În afară de a utiliza funcțiile din biblioteca de funcții a limbajului PHP, vă puteți defini și folosi propriile funcții. Pentru a defini o funcție, respectați modelul următor:

```
function nume_funcție(nume_argument)
{
 // aici se insereaza corpul funcției
}
```

În cadrul modelului, `nume_funcție` este numele funcției, iar `nume_argument` este numele argumentului funcției. În PHP, numele funcțiilor nu prezintă sensibilitate la diferența între majuscule și minuscule; ca atare, `f()` și `F()` reprezintă referiri la aceeași funcție. Cuvântul cheie `function`, numele funcției și lista cu argumente alcătuiesc *antetul funcției*. Termenul de *corp al funcției* se referă la instrucțiunile incluse între parantezele acolade care urmează după antetul funcției. Instrucțiunile din corpul funcției sunt executate atunci când funcția este apelată.

Dacă doriți să definiți o funcție care nu are argumente, puteți omite `nume_argument`; dacă doriți să definiți o funcție cu mai multe argumente, puteți include argumente suplimentare după `nume_argument`, fiecare argument fiind separat de vecinul său prin intermediul unei virgule. Parantezele și numele argumentelor incluse între acestea poartă numele de *listă cu argumente*. Ca exemplu, iată o funcție care calculează aria unui dreptunghi:

```
function calculeaza_arie($inaltime, $latime)
{
 return $inaltime * $latime;
}
```

#### Răspunsuri la test:

- `require("subsol.inc");`
- `require_once("subsol.inc");`

Lista cu argumente a funcției `calculeaza_arie` include argumentele `$latime` și `$inaltime`. Corpul funcției este alcătuit dintr-o singură instrucțiune; cu toate acestea, corpul unei funcții poate conține un număr arbitrar de instrucțiuni. Dacă doriți ca o funcție să returneze o valoare, trebuie să determinați funcția să execute o instrucțiune `return` care furnizează valoarea respectivă. Instrucțiunea `return` determină sistarea executării funcției; nu este necesar ca aceasta să fie ultima instrucțiune fizică din corpul funcției. Dacă definiți o funcție care nu are nici o instrucțiune `return`, funcția va returna valoarea specială `NULL`.

## Apelarea unei funcții definite de utilizator

O funcție definită de utilizator poate fi apelată în același mod ca o funcție încorporată. De exemplu, iată o instrucțiune care apelează funcția `calculeaza_arie`:

```
$arie = calculeaza_arie(2, 4);
```

Valorile *argumentelor efective* – 2 și 4 – le înlocuiesc pe acelea ale *argumentelor formale* din corpul funcției `calculeaza_arie`, care se comportă ca și cum ar fi fost scrisă astfel:

```
return 2 * 4;
```

### Sugestie

În PHP 3, definiția unei funcții trebuie să preceadă linia de program care apelează funcția. În PHP 4, definiția unei funcții poate fi plasată fie anterior liniei de program care apelează funcția, fie după aceasta.

## Terminarea execuției unei funcții

O instrucțiune `return` determină sistarea execuției funcției care o conține. În cazul în care doriți să sistați prelucrarea unui întreg script, puteți invoca funcția `exit()`. Iată un exemplu simplu:

```
function stop()
{
 exit();
}
echo "
Unu...";
echo "
Doi...";
stop();
echo "
Treii!";
```

Rezultatul acestui script include cuvintele *unu* și *doi*, dar nu și cuvântul *trei*. Prin apelarea funcției `stop()` se execută corpul funcției respective; la invocarea funcției `exit()`, execuția scriptului se încheie.

## Funcții recursive

Este posibil ca o funcție din PHP să se auto-apeleze. O funcție care procedează astfel se numește *funcție recursivă*. Totuși, dacă nu ați studiat informatica, este recomandat să nu scrieți funcții recursive. Cu toate acestea, puteți scrie accidental o asemenea funcție, deci este util să știți unele noțiuni referitoare la aceasta.

Programul următor definește și invocă o funcție recursivă simplă:

```
function recursor()
{
 return recursor();
}
$x = recursor();
```

La invocarea funcției `recursor()`, aceasta invocă imediat funcția `recursor()`, care se auto-invocă instantaneu. Astfel, funcția `recursor()` este invocată în mod repetat, până când se produce o eroare cunoscută sub numele de *depășire în sens pozitiv a stivei* (*stack overflow*). Dacă programul dumneavoastră se încheie cu o depășire în sens pozitiv a stivei, o posibilă cauză poate consta într-o recursie incorectă.

## Definirea argumentelor prestabilite

PHP 4 vă permite să definiți funcții cu argumente prestabilite. Dacă invocați o funcție care are un argument prestabilit, dar nu furnizați nici o valoare pentru argumentul respectiv, argumentul ia o valoare prestabilită specificată. Iată un exemplu simplu:

```
function impozit_vanzari($cantitate, $rata=0.0725)
{
 echo "
cantitate=$cantitate";
 echo "
rata=$rata";
 return $suma * $rata;
}

$sumparaturi = 123.45;
echo "
sumparaturi=$sumparaturi";
$impozit = impozit_vanzari($sumparaturi, 0.08);
echo "
impozit=$impozit";

$sumparaturi = 123.45;
echo "
sumparaturi=$sumparaturi";
$impozit = impozit_vanzari($sumparaturi);
echo "
impozit=$impozit";
```

Funcția `impozit_vanzari` preia două argumente: un argument obligatoriu, denumit `$cantitate`, și un argument prestabilit, denumit `$rata`. Dacă apeleți funcția și furnizați un singur argument, valoarea argumentului respectiv se consideră ca fiind valoarea argumentului `$cantitate`, iar valoarea 0.0725 se folosește ca valoare a argumentului `$rata`. Astfel, la prima invocare a funcției, `$rata` are valoarea 0.08,

specificată drept al doilea argument al funcției. Cu toate acestea, la a doua invocare a funcției, `$rata` are valoarea 0.0725, deoarece este specificată valoarea unui singur argument.



### Sfatul specialistului

**Întrebare:** Este posibil ca o funcție să aibă mai multe argumente prestabilite?

**Răspuns:** Da, dar rezultatul este deseori confuz. De exemplu, să considerăm următorul antet de funcție:

```
function doua_rele($timp = 1, $spatiu = 2)
```

- Care trebuie să fie rezultatul dacă este invocată următoarea funcție?

```
x = doua_rele(3);
```

Cu alte cuvinte, argumentul trebuie folosit ca valoare a variabilei `$timp` sau ca valoare a variabilei `$spatiu`? PHP are reguli care controlează asemenea situații, dar regulile respective pot fi derutante. Ca atare, cel mai bine este să evitați problema prin definirea unui singur argument prestabilit și prin poziționarea unui argument prestabilit pe ultima poziție în lista cu argumente.



### Test „la minut”

- Scrieți definiția unei funcții care calculează aria unui cerc, dacă se cunoaște raza acestuia.
- Scrieți definiția unei funcții care are ca date de ieșire blocul `<HEAD>` al unei pagini HTML, incorporând în cadrul unui bloc `<TITLE>` șirul dat ca argument al funcției.

Răspunsuri la test:

```
• function arie_circulara($raza)
{
 return 3.14159 * $raza * $raza;
}

• function bloc_antet($titlu)
{
 echo "<HEAD>\n";
 echo "<TITLE>$titlu</TITLE>\n";
 echo "</HEAD>\n";
}
```

## Variabile și referințe PHP

Variabilele PHP sunt de două tipuri principale:

- Variabile globale
- Variabile locale

Variabilele globale sunt create în exteriorul oricărei funcții, în timp ce variabilele locale sunt create în interiorul unei funcții. Această secțiune descrie variabilele globale și locale, precum și referințele, care constituie o modalitate specială de referire la o variabilă.

### Utilizarea variabilelor globale

Așa cum s-a explicat anterior, variabilele globale sunt declarate în afara oricărei funcții. Variabilele de formular reprezintă un tip important de variabile globale. Cu toate acestea, puteți crea o variabilă globală atribuindu-i acesteia o valoare, atâta timp cât instrucțiunea de atribuire respectivă nu se află în interiorul corpului unei funcții.

Totalitatea locurilor unde este accesibilă o variabilă se numește *domeniu de existență* al variabilei. În mod prestabilit, variabilele globale nu pot fi accesibile din interiorul corpului unei funcții; cu alte cuvinte, domeniul de existență al unei variabile globale nu include corpurile funcțiilor. Dacă doriți să obțineți accesul la o variabilă globală în cadrul unei funcții, puteți extinde domeniul de existență al variabilei prin specificarea numelui variabilei în interiorul unei instrucțiuni `global`. Instrucțiunea `global` are următoarea formă:

```
global variabila1, variabila2, variabila3
```

După cuvântul cheie `global` pot urma una sau mai multe variabile; fiecare variabilă este separată de vecina sa prin intermediul unei virgule. Iată un exemplu care prezintă modul de funcționare a instrucțiunii `global`:

```
function nu_este_global()
{
 echo "
nuglobal: x=$x";
}
function este_global()
{
 global $x;
 echo "
global: x=$x";
}

$x = 1;
nu_este_global();
este_global();
```

Dacă rulați acest script, veți primi următoarele rezultate:

```
nuglobal: x=
global: x=1
```

Rețineți că variabila `$x` primește numele unei valori în afara corpului oricăreia dintre funcții; cu alte cuvinte, `$x` este o variabilă globală. În consecință, variabila `$x` nu se află în cadrul domeniului de existență al funcției `nu_este_global()` și, în consecință, instrucțiunea `echo` din cadrul funcției `nu_este_global()` nu afișează nici o valoare. Cu toate acestea, funcția `este_global()` conține o instrucțiune `global` care extinde domeniul de existență al variabilei `$x`; ca atare, instrucțiunea `echo` din cadrul funcției `este_global()` afișează valoarea variabilei `$x`.

### Utilizarea variabilelor locale și a variabilelor statice

Domeniul de existență, care descrie *unde* este disponibilă o anumită variabilă, reprezintă o importantă caracteristică a variabilelor. O altă caracteristică importantă este *durata de viață*, care descrie *când* este disponibilă o anumită variabilă.

Variabilele globale sunt create atunci când li s-a atribuit o valoare și există pe durata unui program. Spre deosebire de acestea, variabilele locale sunt create la apelarea funcției asociate și sunt distruse la încheierea apelului la funcția respectivă. În consecință, variabilele locale sunt disponibile numai pe durata execuției funcției asociate.

Argumentele funcțiilor constituie un tip important de variabilă locală. Cu toate acestea, puteți crea o variabilă locală prin simpla atribuire a unei valori unei variabile din interiorul unei funcții. Pentru a ilustra deosebirea dintre variabilele locale și cele globale, iată un script care definește o variabilă locală denumită `$x` și o variabilă globală cu același nume:

```
function are_local()
{
 $x = 2;
 echo "
In corpul functiei: x=$x";
}

$x = 1;
echo "
In corpul scriptului: x=$x";
are_local();
echo "
In corpul scriptului: x=$x";
```

În cazul în care rulați acest script, veți primi următoarele rezultate:

```
In corpul scriptului: x=1
In corpul functiei: x=2
In corpul scriptului: x=1
```

Remarcați diferența dintre cele două variabile, chiar dacă numele variabilelor este același. Domeniul de existență al variabilei globale `$x` nu se extinde în interiorul corpului funcției `are_local()`, iar domeniul de existență al variabilei locale `$x` nu se extinde dincolo de corpul funcției respective. Cu alte cuvinte, domeniile de existență ale celor două variabile sunt complet distincte și, ca atare, PHP nu poate confunda valorile variabilelor respective.

Uneori, doriți ca o variabilă locală să-și păstreze valoarea de la un apel al funcției asociate la altul. Altfel spus, nu doriți ca variabila să fie distrusă la încheierea apelului la funcție. Puteți folosi instrucțiunea `static` pentru a specifica acest comportament. Forma instrucțiunii `static` este similară cu aceea a instrucțiunii `global`, cu excepția utilizării cuvântului cheie `static` în locul cuvântului cheie `global`. O variabilă afișată într-o instrucțiune `static` este cunoscută sub numele de *variabilă locală statică*, sau, mai concis, *variabilă statică*. Iată un exemplu care prezintă modul de utilizare a unei variabile statice:

```
function nu_este_static();
{
 $x = $x + 1;
 echo "
x=$x";
}

function este_static();
{
 static $x;
 $x = $x + 1;
 echo "
x=$x";
}

nu_este_static();
nu_este_static();
nu_este_static();

este_static();
este_static();
este_static();
```

Dacă rulați acest script, veți primi următoarele rezultate:

```
x=1
x=1
x=1
x=1
x=2
x=3
```

Observați că variabila locală `$x`, definită în cadrul funcției `nu_este_static()`, este creată din nou la fiecare apelare a funcției. În consecință, valoarea sa este întotdeauna afișată ca fiind egală cu 1. Prin contrast, variabila statică `$x`, definită în cadrul funcției `este_static()`, își păstrează valoarea de la un apel al funcției la următorul; ca atare, valoarea sa crește de fiecare dată când este executată funcția. De asemenea, rețineți că domeniile de existență ale variabilei locale `$x` și ale variabilei statice `$x` sunt distincte; în consecință, valorile celor două variabile sunt diferite, chiar dacă variabilele au același nume.

## Utilizarea referințelor (PHP 4)

În mod prestabilit, argumentele transferate unei funcții PHP sunt transmise *prin valoare*, ceea ce înseamnă că valorile argumentelor sunt copiate și funcțiile utilizează copii ale valorilor argumentelor lor, nu argumentele în sine. Ca o consecință, o funcție PHP nu poate modifica valorile argumentelor sale. Acest fapt este în general util, deoarece astfel funcțiile devin mai ușor de înțeles dacă se limitează la a returna o valoare, fără a modifica valori.

Totuși, puteți stabili ca o funcție să aibă posibilitatea de a modifica valoarea unui argument, specificând ca argumentul să fie transferat *prin referință*. Când un argument este transferat prin referință, valoarea sa nu este copiată; funcția lucrează cu valoarea argumentului și are libertatea de a modifica acea valoare. Pentru a specifica faptul că un argument urmează a fi transferat prin referință, argumentul va fi prefixat cu un caracter ampersand (&). Puteți atașa acest prefix la argument în antetul funcției sau în apelul la funcție.

Iată un exemplu care prezintă apelul prin valoare și apelul prin referință:

```
function prin_valoare($x)
{
 $x = 1;
}

function prin_referinta(&$x)
{
 $x = 1;
}

$y = 0;
prin_valoare($y);
echo "
\$y=$y";

$y = 0;
prin_valoare(&$y);
echo "
\$y=$y";

$y = 0;
prin_referinta($y);
echo "
\$y=$y";
```

Dacă rulați acest script, veți obține următoarele date de ieșire:

```
$y=0
$y=1
$y=1
```

Rețineți că scriptul conține două funcții, și anume `prin_valoare()` și `prin_referinta()`. Fiecare funcție preia un singur argument, denumit `$x`. Antetul funcției `prin_referinta()` specifică faptul că argumentul său este transferat prin referință; argumentul funcției `prin_valoare()` este transferat prin valoare, în conformitate cu practica prestabilită folosită în limbajul PHP.



Fiecare funcție încearcă să modifice valoarea argumentului său. Primul paragraf al programului invocă funcția `prin_valoare()`, transferând argumentul prin valoare. În consecință, funcția lucrează cu o copie a argumentului său, iar valoarea variabilei `$y` nu se modifică.

Cel de-al doilea paragraf al programului invocă de asemenea funcția `prin_valoare()`; cu toate acestea, folosește un caracter ampersand pentru a determina transferul prin referință al valorii variabilei `$y`. În consecință, funcția modifică valoarea argumentului său, care se transformă din 0 în 1.

Cel de-a treilea paragraf al programului invocă funcția `prin_referinta()`. Antetul funcției respective folosește un caracter ampersand pentru a specifica faptul că valoarea argumentului său este transferată prin referință. În consecință, funcția modifică valoarea argumentului său, care se transformă din 0 în 1.



## Sfatul specialistului

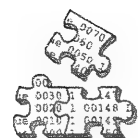
**Întrebare:** De ce se folosesc referințele sau apelurile prin referință?

**Răspuns:** Prin utilizarea referințelor se evită suprasarcina de copiere a valorilor argumentelor și implicit se obține o viteză mai mare de execuție a programului. Cu toate acestea, programele devin astfel mai dificil de înțeles, iar referințele sau apelurile prin referință pot cauza erori de program. Cel mai indicat este să evitați referințele, acolo unde este posibil, și să definiți funcții care returnează valori, și nu funcții care modifică valorile propriilor argumente. Cu toate acestea, este important să înțelegeți noțiunile privind referințele, astfel încât să puteți lucra cu programe scrise de programatori care nu respectă aceste sfaturi.



## Test „la minut”

- Scrieți instrucțiunea care este necesară într-o funcție pentru a se putea obține accesul la o variabilă denumită `$greutate` creată în afara funcției.
- Scrieți instrucțiunea care determină variabila locală `$contor` să-și păstreze valoarea la mai multe invocări ale funcției asociate.



## Proiect 7-1: Revenire la formularul cu persoane de contact

În cadrul acestui proiect, veți crea din nou formularul HTML și scriptul PHP intitulat „Proiect 6-2”. Formularul permite unui utilizator să introducă date personale de categoria celor folosite într-o agendă cu adrese de e-mail.

Răspunsuri la test:

- `global $greutate;`
- `static $contor;`

Scriptul PHP validează datele de intrare introduse de utilizator, asigurându-se că au fost introduse date în câmpurile obligatorii.

## Scopurile proiectului

- Prezentarea modului de funcționare a instrucțiunii `include`
- Prezentarea modului de definire și creare a unei funcții
- Prezentarea tehnicii racordurilor de program

## Pas cu pas

1. Plasati următorul script PHP într-un fișier denumit `p-7-1.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php

include "p-7-1.inc";

function validare_formular()
{
 global $porecla, $prenume, $nume, $email;

 $erori = 0;
 if(!trim($porecla))
 {
 echo "
Porecla este obligatorie.";
 $erori++;
 }

 if(!trim($prenume))
 {
 echo "
Prenumele este obligatoriu.";
 $erori++;
 }

 if(!trim($nume))
 {
 echo "
Numele este obligatoriu.";
 $erori++;
 }

 if(!trim($email))
 {
 echo "
Adresa primara de e-mail "
 . "este obligatorie.";
 $erori++;
 }

 switch($erori)
 {
 case 0:
 return TRUE;

 case 1:
 echo "

Va rugam folositi butonul ";
 echo "Back al browserului dumneavoastra ";
 echo "pentru a reveni la formular.";
 }
}
```



```

 echo "corectati eroarea si ";
 echo "reexpediati formularul.";
 return FALSE;

 default:
 echo "

Va rugam folositi butonul ";
 echo "Back al browserului dumneavoastra ";
 echo "pentru a reveni la formular, ";
 echo "corectati erorile si ";
 echo "reexpediati formularul.";
 return FALSE;
 }
}

function actualizare_baza_de_date()
{
 echo "
Actualizeaza baza de date....";
}

$ok = validare_formular();
if ($ok)
 actualizare_baza_de_date();

?>

</BODY>
</HTML>

```

2. Plasati urmatorul text HTML într-un fișier denumit p-7-1.inc și încărcati acest fișier în serverul dumneavoastră, plasându-l în același catalog ca și fișierul p-7-1.php:

```

<HTML>
<HEAD>
<TITLE>Proiect 7-1</TITLE>
</HEAD>
</BODY>

```

3. Plasati urmatorul text HTML într-un fișier denumit p-7-1.html și încărcati acest fișier în serverul dumneavoastră, plasându-l în același catalog ca și fișierul p-7-1.php:

```

<HTML>
<HEAD>
<TITLE>Proiect 7-1</TITLE>
</HEAD>
<BODY>
<!-- Fișier p-7-1.html -->
<FORM METHOD="POST" ACTION="p-7-1.php">
<H1>Informatii privind persoana de contact</H1>
<TABLE>
<TR>
<TD>Porecla:</TD>
<TD><INPUT TYPE="TEXT" NAME="porecla"></TD>
</TR>
<TR>
<TD>Titlu:</TD>
<TD><INPUT TYPE="TEXT" NAME="titlu"></TD>
</TR>

```

```

<TR>
<TD>Prenume:</TD>
<TD><INPUT TYPE="TEXT" NAME="prenume"></TD>
</TR>
<TR>
<TD>Prenumele tatalui:</TD>
<TD><INPUT TYPE="TEXT" NAME="prenume_tata"></TD>
</TR>
<TR>
<TD>Nume:</TD>
<TD><INPUT TYPE="TEXT" NAME="Nume"></TD>
</TR>
<TR>
<TD>Adresa de e-mail principala:</TD>
<TD><INPUT TYPE="TEXT" NAME="email"></TD>
<TD WIDTH="20"> </TD>
<TD>Adresa de e-mail secundara:</TD>
<TD><INPUT TYPE="TEXT" NAME="emailsecundar"></TD>
</TR>
<TR>
<TD>Numele companiei:</TD>
<TD><INPUT TYPE="TEXT" NAME="nume_companie"></TD>
</TR>
<TR>
<TD>Adresa firmei:</TD>
<TD><INPUT TYPE="TEXT" NAME="adresa_firmei1"></TD>
<TD WIDTH="20"> </TD>
<TD>Adresa la domiciliu:</TD>
<TD><INPUT TYPE="TEXT" NAME="adresa_acasa"></TD>
</TR>
<TR>
<TD></TD>
<TD><INPUT TYPE="TEXT" NAME="adresa_firmei2"></TD>
</TR>
<TR>
<TD>Oras:</TD>
<TD><INPUT TYPE="TEXT" NAME="oras_birou"></TD>
<TD WIDTH="20"> </TD>
<TD> </TD>
<TD><INPUT TYPE="TEXT" NAME="oras_acasa"></TD>
</TR>
<TR>
<TD>Stat:</TD>
<TD><INPUT TYPE="TEXT" NAME="stat_birou"></TD>
<TD WIDTH="20"> </TD>
<TD> </TD>
<TD><INPUT TYPE="TEXT" NAME="stat_acasa"></TD>
</TR>

```

```

<TR>
 <TD>Cod postal:</TD>
 <TD><INPUT TYPE="TEXT" NAME="cod_birou"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD><INPUT TYPE="TEXT" NAME="cod_acasa"></TD>
</TR>

<TR>
 <TD>Telefon:</TD>
 <TD><INPUT TYPE="TEXT" NAME="telefon_birou"></TD>
 <TD WIDTH="20"> </TD>
 <TD> </TD>
 <TD><INPUT TYPE="TEXT" NAME="telefon_acasa"></TD>
</TR>

<TR>
 <TD>Data nasterii:</TD>
 <TD><INPUT TYPE="TEXT" NAME="data_nastere"></TD>
</TR>

<TR>
 <TD>Numele sotului/sotiei:</TD>
 <TD><INPUT TYPE="TEXT" NAME="nume_sot"></TD>
 <TD WIDTH="20"> </TD>
 <TD>Numele copiilor:</TD>
 <TD><INPUT TYPE="TEXT" NAME="copii"></TD>
</TR>

<TR>
 <TD>Ziua nuntii:</TD>
 <TD><INPUT TYPE="TEXT" NAME="zi_nunta"></TD>
</TR>
</TABLE>

<INPUT TYPE="SUBMIT" VALUE="Trimite">

<INPUT TYPE="RESET" VALUE="Sterge datele">

</FORM>
</BODY>
</HTML>

```

4. Alocăți un timp studiului scriptului PHP, acordând o atenție specială modificărilor operate în proiectul inițial, prin care au fost incluse un fișier de includere și anumite funcții. În particular, observați că funcția `actualizare_baza_de_date()` conține o singură instrucțiune `echo`. O asemenea funcție, cunoscută sub numele de *racord*, este folosită în faza de dezvoltare a programului, astfel încât programul să poată fi rulat și testat chiar înainte de a fi fost scris în totalitate. Ulterior, corpul funcției `actualizare_baza_de_date()` va fi rafinat, urmând să conțină instrucțiuni care realizează efectiv actualizarea unei baze de date.

5. Orientați un browser Web spre adresa URL a fișierului HTML încărcat anterior. Ecranul browserului trebuie să fie asemănător celui prezentat în continuare. Introduceți valori pentru mai multe câmpuri și apoi executați clic pe butonul Trimite.

6. La executarea scriptului, acesta verifică dacă porecla, prenumele, numele și adresa de e-mail sunt prezente. Dacă oricare dintre aceste câmpuri lipsește, scriptul afișează un mesaj de eroare. Un rezultat caracteristic este prezentat în continuare.

## Test de evaluare

1. Scrieți o instrucțiune care invocă funcția `test()`, transferând valorile 1 și 2 ca argumente.
2. Scrieți o instrucțiune care invocă funcția `live()`, transferând ca argumente valorile 1 și 2; asigurați-vă că nu se vor genera mesaje de eroare în timpul execuției funcției.

3. Scrieți o instrucțiune care include conținutul fișierului `antet.php` ca parte a scriptului curent.
4. Scrieți definiția unei funcții numite `patrat()`, care calculează aria unui pătrat, dacă este dată lungimea unei laturi a pătratului.
5. Scrieți o definiție a unei funcții denumite `contor()`, care incrementează și returnează valoarea unei variabile locale statice.

## Modulul 8:

# Utilizarea tablourilor

### Scopuri

- Învățați să creați un tablou
- Învățați să parcurgeți iterativ un tablou
- Învățați să lucrați cu funcții listă
- Învățați să sortați un tablou
- Învățați să folosiți tablouri în lucrul cu variabilele dintr-un formular

În Modulul 2 a fost prezentată noțiunea de tablou, un tip de variabilă special, care poate avea mai multe valori asociate. În cadrul acestui modul, veți învăța mai multe despre utilizarea tablourilor.

### Crearea tablourilor

Puteți crea un tablou PHP în două moduri principale. Puteți atribui o valoare unei variabile dintr-un tablou sau puteți invoca funcția `array()`. În această secțiune vor fi explicate ambele metode de creare a unui tablou.

#### Crearea unui tablou folosind atribuiri

Modalitatea cea mai simplă de a crea un tablou este de a atribui o valoare unei variabile de tip tablou. Iată un exemplu simplu în acest sens:

```
$limbaje[] = "Perl";
$limbaje[] = "PHP";
$limbaje[] = "Python";
```

Parantezele drepte care urmează după numele variabilei indică limbajului PHP că variabila `$limbaje` este o variabilă de tip tablou. PHP stochează în mod automat valorile atribuite tabloului în celule numerotate succesiv, începând de la `$limbaje[0]`. Iată conținutul tabloului, așa cum rezultă din cele trei instrucțiuni de atribuire:

```
0 => Perl
1 => PHP
2 => Python
```

Simbolul `=>` arată că unei valori îi este asociată o cheie. În acest caz, cheia 0 este asociată cu valoarea "Perl", cheia 1 este asociată cu valoarea "PHP", iar cheia 2 este asociată cu valoarea "Python". Puteți asimila tabloul cu un tabel, caz în care exemplul anterior este reprezentat astfel:

```
0 Perl
1 PHP
2 Python
```

Dacă doriți, puteți scrie o instrucțiune de atribuire care specifică o valoare cheie, astfel încât să puteți asocia o valoare cu un anumit element al tabloului. Să luăm în considerare următorul exemplu:

```
$limbaje[0] = "Perl";
$limbaje[1] = "PHP";
$limbaje[] = "Python";
```

Observați că instrucțiunea finală de atribuire nu include nici o valoare de index. PHP asociază valoarea "Python" cu următorul element consecutiv al tabloului:

```
0 => Perl
1 => PHP
2 => Python
```

Elementele unui tablou nu trebuie asociate unor chei consecutive. Iată un exemplu care demonstrează acest fapt:

```
$limbaje[10] = "Perl";
$limbaje[20] = "PHP";
$limbaje[] = "Python";
```

Aceste instrucțiuni de atribuire determină următorul conținut al tabloului:

```
10 => Perl
20 => PHP
21 => Python
```

Remarcați, similar cu exemplul anterior, că valoarea "Python" este asociată următorului element consecutiv din tablou.

Așa cum s-a explicat în Modulul 2, PHP nu vă obligă să folosiți numere întregi pe post de chei. Puteți crea un *tablou asociativ*, cunoscut și sub numele de *tablou indexat cu șiruri*, prin specificarea drept chei a unor șiruri:

```
$limbaje['PHP'] = "Ridicat";
$limbaje['Python'] = "Mediu";
$limbaje['Perl'] = "Redus";
```

Prin instrucțiunile de mai sus, următoarele chei au fost atribuite următoarelor valori:

```
PHP => Ridicat
Python => Mediu
Perl => Redus
```

Semnificația utilizării pe post de chei a unor valori întregi consecutive este aceea că puteți folosi o buclă `for` simplă pentru a parcurge iterativ tabloul, cu alte cuvinte, pentru a examina valorile fiecăruia dintre elementele sale. Veți învăța mai multe despre parcurgerea iterativă a tablourilor în secțiunea următoare, intitulată „Utilizarea funcției `array()`”.

## Utilizarea funcției `array()`

Dincolo de utilizarea instrucțiunilor de atribuire, cealaltă modalitate principală de creare a unui tablou PHP constă în utilizarea funcției `array()`. Iată un exemplu simplu, care creează un tablou având drept chei valori întregi consecutive:

```
$limbaje = array("Perl", "PHP", "Python");
```

Această instrucțiune creează un tablou care conține următoarele asocieri:

```
0 => Perl
1 => PHP
2 => Python
```

Dacă doriți să asociați unei valori o anumită cheie, puteți folosi operatorul `=>`, astfel:

```
$limbaje = array(10=>"Perl", "PHP", "Python");
```

Această instrucțiune creează următorul tablou:

```
10 => Perl
11 => PHP
12 => Python
```

Ca și în cazul utilizării unei instrucțiuni de atribuire pentru crearea unui tablou, valorile cheilor nu trebuie să fie consecutive și nici măcar întregi:

```
$limbaje = array("PHP"=>"Ridicat", "Python"=>"Mediu", "Perl"=>"Redus");
```

Această instrucțiune creează următorul tablou:

```
PHP => Ridicat
Python => Mediu
Perl => Redus
```



### Sfatul specialistului

**Întrebare:** În cadrul altor limbaje de programare, este posibilă crearea de tablouri multi-dimensionale. Se poate proceda astfel și în PHP?

**Răspuns:** Un tablou multi-dimensional poate fi asimilat, pur și simplu, unui tablou ale cărui celule conțin valori ale unui tablou; cu alte cuvinte, un tablou de tablouri. De exemplu, să presupunem că doriți să caracterizați ușurința de învățare și popularitatea mai multor limbaje de scripting. Puteți reprezenta datele într-un tabel, în următoarea formă:

Limbaj	Ușurință în învățare	Popularitate
PHP	simplu	popular
Python	simplu	nepopular
Perl	dificil	popular

Puteți reprezenta aceste date sub forma unui tablou PHP denumit `$date`, prin scrierea următoarelor instrucțiuni:

```
$date["PHP"] = array("simplu", "popular");
$date["Python"] = array("simplu", "nepopular");
$date["PHP"] = array("difícil", "popular");
```

Sau puteți scrie următoarea instrucțiune:

```
$date = array("PHP"=>array("simplu", "popular"),
"Python"=>array("simplu", "nepopular"),
"PHP"=>array("difícil", "popular"));
```



### Test „la minut”

- Scrieți instrucțiuni de atribuire PHP care atribuie numele principalelor culori aditive (roșu, verde și albastru) unor chei numerotate consecutiv ale unui tablou denumit `$culoare`.
- Scrieți instrucțiuni de atribuire PHP care asociază valorile 100, 1000 și 1000000 cheilor "suta", "mie" și "milion" în cadrul unui tablou denumit `$marime`.

## Parcursarea iterativă a unui tablou

Când ați stocat date într-un tablou, puteți obține acces la valoarea unui element al tabloului sau îi puteți modifica valoarea prin intermediul cheii asociate elementului. De exemplu, să presupunem că folosiți următoarele instrucțiuni pentru a crea un tablou:

```
$x[0] = 1;
$x[1] = 10;
$x[2] = 1000;
```

Puteți obține acces la valoarea asociată cheii 1 prin intermediul unei instrucțiuni ca aceasta:

```
$y = 3 * $x[1];
```

Similar, puteți modifica valoarea asociată cheii 2 prin intermediul unei instrucțiuni ca aceasta:

```
$x[2] = 100;
```

Răspunsuri la test:

- `$culoare[] = "roșu";`  
`$culoare[] = "verde";`  
`$culoare[] = "albastru";`
- `$marime["suta"] = 100;`  
`$marime["mie"] = 1000;`  
`$marime["milion"] = 1000000;`

Uneori, în loc de a obține accesul la un singur element al unui tablou sau de a-l modifica, doriți să obțineți accesul la mai multe elemente ale tabloului. De exemplu, să presupunem că doriți să determinați dacă în tablou există o anumită cheie sau valoare. Sau să presupunem că tabloul reprezintă salarii și doriți să măriți fiecare valoare cu 10 procente. Operații de acest gen implică parcurgerea iterativă a tabloului sau, altfel spus, accesul la fiecare element al tabloului. Această secțiune explică modul de iterație prin tablouri secvențiale și non-secvențiale.

## Parcursarea iterativă a unui tablou secvențial

Un tablou ale cărui chei sunt valori întregi consecutive se numește *tablou secvențial*. În general, valoarea cea mai mică a unei chei dintr-un tablou secvențial este zero; totuși, puteți crea un tablou secvențial folosind valoarea unu sau orice altă valoare întreagă ca valoare minimă a cheii.

În cazul în care cunoașteți valoarea minimă a cheii unui tablou secvențial, puteți parcurge iterativ tabloul folosind o buclă `for`. Pentru aceasta, inițializați variabila de buclă la valoarea cea mai redusă a cheii. Folosiți funcția `count()` pentru a forma expresia de test a buclei. Funcția `count()` returnează numărul elementelor dintr-un tablou. Iată un exemplu simplu de iterație într-un tablou:

```
$limbaje = array(0=>"Perl", 1=>"PHP", 2=>"Python");
$limita = count($limbaje);
for ($i = 0; $i < $limita; $i++)
{
 echo "
$i => $limbaje[$i]";
}
```

Prima instrucțiune creează tabloul. Cea de-a doua instrucțiune obține numărul elementelor din tablou. Instrucțiunea `for` folosește variabila buclă `$i` pentru a parcurge iterativ tabloul; corpul instrucțiunii include o instrucțiune `echo` care afișează cheia și valoarea fiecărui element din tablou. Datele de ieșire se prezintă astfel:

```
0 => Perl
1 => PHP
2 => Python
```

## Căutarea într-un tablou secvențial

Pentru a examina modul de utilizare a iterațiilor, să ne concentrăm asupra problemei de a determina dacă un tablou conține o anumită valoare. Iată un exemplu:

```
$limbaje = array(0=>"Perl", 1=>"PHP", 2=>"Python");
$scuta = "PHP";
// se caută în tabloul $limbaje valoarea $scuta
$limita = count($limbaje);
for ($i = 0; $i < $limita; $i++)
{
 echo "
Determinarea unei identități cu $limbaje[$i]";
}
```

```

if ($scuta == $limbaje[$i])
{
 "
$scuta este un limbaj aprobat.";
}
}

```

Prima instrucțiune creează tabloul în care se va căuta. Desigur, într-o aplicație iterativă reală, tabloul nu va fi inițializat cu valori literale imediat anterior operației de căutare. Într-o aplicație reală, conținutul tabloului este supus la variații.

Cea de-a doua instrucțiune atribuie valoarea "PHP" variabilei \$scuta; în exemplu, se caută în tablou valoarea stocată în variabila \$scuta. După comentariu, următoarea instrucțiune obține numărul elementelor din tablou și stochează această valoare în variabila \$limita. Instrucțiunea for funcționează ca mai înainte; de data aceasta însă, corpul său conține alte instrucțiuni și se execută o altă operație. O instrucțiune echo afișează valoarea fiecărui element al tabloului pe măsură ce iterația avansează. Instrucțiunea if testează fiecare element și afișează un mesaj dacă valoarea elementului este una și aceeași cu valoarea variabilei \$scuta. Iată rezultatul rulării exemplului:

```

Determinarea unei identități cu Perl
Determinarea unei identități cu PHP
PHP este un limbaj aprobat.
Determinarea unei identități cu Python

```

## Instrucțiunea break

Observați că iterația continuă chiar și după stabilirea unei identități. Când se caută într-un tablou, execuția căutării poate fi sistată după găsirea elementului dorit; continuarea iterației în tablou nu face decât să irosească resursele calculatorului, fără a afecta rezultatele operației. Pentru a opri execuția unei iterații, puteți folosi instrucțiunea break, care determină încheierea imediată a buclei care o conține. Iată cum se poate revizui exemplul anterior, astfel încât să includă o instrucțiune break:

```

for ($i = 0; $i < $limita ; $i++)
{
 echo "
Determinarea unei identități cu $limbaje[$i]";
 if ($scuta == $limbaje[$i])
 {
 "
$scuta este un limbaj aprobat.";
 break;
 }
}

```

Acum, după stabilirea unei identități, instrucțiunea break provoacă sistarea buclei for. Iată datele de ieșire rezultate, care acum omit examinarea inutilă a elementului tabloului asociat cu limbajul "Python":

```

Determinarea unei identități cu Perl
Determinarea unei identități cu PHP
PHP este un limbaj aprobat.

```

## Instrucțiunea continue

O instrucțiune corelată cu instrucțiunea break este continue. Instrucțiunea continue sistează iterația curentă a buclei, determinând evaluarea imediată a expresiilor de incrementare și de test. Ca un exemplu, să presupunem că doriți să căutați în tabloul \$limbaje pentru a determina numărul limbajelor care au nume scurte, adică nume alcătuite din maximum patru caractere. Iată un exemplu care execută această prelucrare a datelor:

```

$limbaje = array(0=>"Perl", 1=>"PHP", 2=>"Python");
// Numara numele scurte
$scurt = 0;
$limita = count($limbaje);
for ($i = 0; $i < $limita; $i++)
{
 $n = strlen($limbaje[$i]);
 echo "
$limbaje[$i] are $n caractere lungime.";
 if ($n > 4) continue;
 $scurt++;
}
echo "
Au fost gasite $scurt limbaje cu nume scurte.";

```

O instrucțiune de atribuire stabilește valoarea inițială a variabilei \$scurt, folosită pentru a număra numele scurte găsite, la zero. Instrucțiunea for se aseamănă celor folosite anterior. Corpul acestei instrucțiuni diferă, desigur, de cele folosite anterior. Valoarea variabilei \$n este stabilită ca fiind egală cu numărul caracterelor care compun numele limbajului, folosind funcția strlen(), care calculează lungimea unui șir. Dacă instrucțiunea if stabilește că elementul curent al tabloului face referire la un limbaj cu nume lung, se execută instrucțiunea continue. Instrucțiunea continue determină trecerea iterației la următorul element din tablou; dacă nu au mai rămas elemente în tablou, bucla for își încheie execuția. La finalizarea iterației, o instrucțiune echo afișează numărul numelor scurte de limbaje găsite în tablou. Iată rezultatul:

```

Perl are 4 caractere lungime.
PHP are 3 caractere lungime.
Python are 6 caractere lungime.
Au fost gasite 2 limbaje cu nume scurte.

```

## Parcurea iterativă a unui tablou non-secvențial

În unele limbaje de programare, parcurea unui tablou non-secvențial este o operație complicată. Cu toate acestea, PHP 4 include o instrucțiune foreach care simplifică iterațiile de acest gen. Instrucțiunea foreach are următoarea formă generală:

```

foreach ($tablou as $cheie => $valoare) { corp }

```

Instrucțiunea parcurge în mod iterativ tabloul denumit \$tablou, stabilind valori adecvate pentru valorile variabilelor \$cheie și \$valoare aferente fiecărui element al tabloului. Iată un exemplu simplu:



```
$limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
// parcurge iterativ tabloul
foreach ($limbaje as $index => $limbaj)
{
 echo "
$index => $limbaj";
}
```

Remarcați că instrucțiunea echo face pur și simplu referire la valorile variabilelor \$index și \$limbaj, cărora li se atribuie în mod automat valorile cheii, respectiv elementului curent. Iată datele de ieșire ale exemplului:

```
10 => Perl
20 => HP
21 => Python
```

## ? Sfatul specialistului

**Întrebare:** Cum pot parcurge în mod iterativ un tablou multi-dimensional?

**Răspuns:** Pentru a răspunde la această întrebare, să parcurgem iterativ un tablou multi-dimensional care conține următoarele date:

Limbaj	Ușurință în învățare	Popularitate
PHP	simplu	popular
Python	simplu	nepopular
Perl	difficil	popular

Tabloul poate fi creat folosind următoarele instrucțiuni PHP:

```
$date["PHP"] = array("simplu", "popular");
$date["Python"] = array("simplu", "nepopular");
$date["Perl"] = array("difficil", "popular");
```

Pentru a parcurge tabloul în mod iterativ, folosiți instrucțiuni foreach imbricate:

```
foreach ($date as $limbaj => $valoare)
{
 foreach ($valoare as $cheie => $caracteristica)
 {
 echo "
$limbaj: $caracteristica";
 }
}
```

Instrucțiunea foreach exterioară obține tabloul asociat cu fiecare limbaj; instrucțiunea foreach interioară parcurge iterativ caracteristicile limbajului. Iată rezultatul:

```
PHP: simplu
PHP: popular
Python: simplu
Python: nepopular
Perl: difficil
Perl: popular
```



## Test „la minut”

- Care este instrucțiunea folosită pentru a parcurge în mod iterativ un tablou non-secvențial?
- Care este funcția ce returnează numărul elementelor dintr-un tablou?
- Care este instrucțiunea folosită pentru a sista iterația curentă a unei bucle?
- Care este instrucțiunea folosită pentru terminarea imediată a unei bucle?

## Lucrul cu funcții listă

În afară de modalitățile de parcurgere iterativă a tablourilor, PHP oferă numeroase funcții care vă permit traversarea tablourilor, deplasându-vă înainte sau înapoi, după dorință. Prima dintre aceste funcții este `current()`, care returnează valoarea curentă al tabloului. Funcția `current()` folosește un pointer intern de tablou pe care PHP îl creează pentru fiecare tablou. Când creați un tablou, pointerul intern de tablou este configurat astfel încât să facă referire la primul element al tabloului. Funcțiile `next()` și `prev()` modifică pointerul intern al tabloului și se pot folosi alături de funcția `current()` pentru a parcurge un tablou. Funcția `next()`, așa cum îi arată și numele\*, modifică pointerul intern al tabloului astfel încât acesta să facă referire la următorul element, în timp ce funcția `prev()` modifică pointerul intern al tabloului astfel încât acesta să facă referire la elementul anterior.

Iată un exemplu care prezintă modul de operare al funcțiilor menționate:

```
$limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
$curent = current($limbaje);
echo "
funcția current() a returnat $curent";
$urmator = next($limbaje);
echo "
funcția next() a returnat $urmator";
$urmator = next($limbaje);
echo "
funcția next() a returnat $urmator";
$anterior = prev($limbaje);
echo "
funcția prev() a returnat $anterior";
```

Iată datele de ieșire ale exemplului:

```
funcția current() a returnat Perl
funcția next() a returnat PHP
funcția next() a returnat Python
funcția prev() a returnat PHP
```

Observați modul în care sunt utilizate funcțiile `prev()` și `next()` pentru deplasarea înapoi, respectiv înainte, în interiorul tabloului.

Răspunsuri la test:

- `foreach`
- `count()`
- `continue`
- `break`

\* În limba engleză *next* – următor. (N.T.)

## Funcția key()

Funcția `key()` este corelată cu funcția `current()`. Dacă funcția `current()` returnează valoarea asociată elementului curent, funcția `key()` returnează cheia asociată elementului curent. Iată un exemplu care ilustrează modul de operare al acestei funcții:

```
$limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
$current = current($limbaje);
$scheie = key($limbaje);
echo "
funcția current() a returnat $current";
echo "
funcția key() a returnat $scheie";
```

Datele de ieșire ale exemplului sunt următoarele:

```
funcția current() a returnat Perl
funcția key() a returnat 10
```

## Funcția each()

O altă funcție utilă în parcurgerea tablourilor este `each()`. Funcția `each()` returnează următoarea pereche cheie-valoare din tabloul specificat. Perechea cheie-valoare este returnată sub forma unui tablou asociativ cu patru elemente, după cum urmează:

Cheie	Valoare
0	Componenta cheie a perechii cheie-valoare curentă
1	Componenta valoare a perechii cheie-valoare curentă
"key"	Componenta cheie a perechii cheie-valoare curentă
"value"	Componenta valoare a perechii cheie-valoare curentă

Observați că puteți folosi valoarea cheie „0” sau „cheie” pentru a obține accesul la componenta *cheie* a perechii cheie-valoare; similar, puteți folosi valoarea cheie „1” sau „valoare” pentru a obține accesul la componenta *valoare* a perechii cheie-valoare. Iată un scurt exemplu care ilustrează modul de operare a funcției `each()`:

```
$limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
$fiecare = each($limbaje);
$zero = $each[0];
$unu = $each[1];
$scheie = $each['key'];
$valoare = $each['value'];
echo "
zero=$zero";
echo "
unu=$unu";
echo "
cheie=$scheie";
echo "
valoare=$valoare";
```

Datele de ieșire ale acestui exemplu sunt următoarele:

```
zero=10
unu=Perl
cheie=10
valoare=Perl
```

## Funcția list()

O altă funcție utilă în lucrul cu tablouri este funcția `list()`, care vă permite să atribuiți valori la numeroase variabile în cadrul unei instrucțiuni. Funcția `list()` este deseori folosită cu funcția `each()`, deoarece funcția `list()` facilitează accesul separat la cheia și la valoarea returnate de funcția `each()`. Forma generală de utilizare a funcției `list()` este următoarea:

```
list($var1, $var2, ..., $varn) = valoare_tablou;
```

Fiecare dintre variabilele specificate, de la `$var1` la `$varn`, primește o valoare din tabloul `valoare_tablou`. Într-un fel, funcția `list()` este opusă funcției `array()`, deoarece funcția `list()` împarte un tablou într-o serie de valori scalare, în timp ce funcția `array()` construiește un tablou dintr-o serie de valori scalare.

Iată un exemplu care ilustrează modul de utilizare a funcției `list()`:

```
$limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
list($scheie, $valoare) = each($limbaje);
echo "
cheie=$scheie, valoare=$valoare";
$urmator = next($limbaje);
echo "
urmator=$urmator";
```

De asemenea, iată datele de ieșire ale exemplului:

```
cheie = 10, valoare = Perl
urmator = Python
```



### Test „la minut”

- Care este funcția ce vă permite să atribuiți valori mai multor variabile simultan?
- Care este funcția ce returnează componenta cheie a unei perechi cheie-valoare asociată elementului curent al tabloului?
- Care este funcția ce mărește valoarea unui pointer intern de tablou?



### Sfatul specialistului

**Întrebare:** PHP mai conține și alte funcții pentru lucrul cu tablourile?

**Răspuns:** Da. PHP conține peste 40 de funcții pentru lucrul cu tablourile, cu mult mai multe decât pot fi descrise în acest capitol. De exemplu, funcția `array_search()` facilitează căutarea într-un tablou. Pentru informații despre această funcție și despre alte funcții utilizate în lucrul cu tablouri, examinați documentația PHP în variantă electronică, la adresa <http://www.php.net>.

Răspunsuri la test:

- `list()`
- `key()`
- `next()`

## Sortarea tablourilor

Sortarea reprezintă o altă operație frecvent aplicată tablourilor. PHP furnizează o suită de funcții care facilitează sortarea unui tablou. De exemplu, un tablou poate fi creat după cum urmează:

```
$limbaje = array(10=>"Perl", 20=>"PHP", 21=>"Python");
```

Apoi, doriți să sortați tabloul în funcție de numele limbajului de programare. Pentru aceasta, pur și simplu invocați funcția `sort()`:

```
sort($limbaje);
```

După sortare, conținutul tabloului apare așa cum se poate vedea mai jos:

```
0 => PHP
1 => Perl
2 => Python
```

Observați că *secvența de sortare* sau *secvența de aranjare* (cum este numită uneori) este sensibilă la diferența între majuscule și minuscule. Deoarece litera *H* mare este sortată anterior literei *e* mic, PHP apare înainte de Python în datele de ieșire sortate.

Tabelul 8-1 prezintă pe scurt funcțiile de sortare ale limbajului PHP, inclusiv rezultatul aplicării fiecărei funcții tabloului folosit în exemplul anterior.

**Tabelul 8-1.** Un sumar al funcțiilor de sortare ale limbajului PHP

Operație	Funcție	Rezultat
Sortarea unui tablou în funcție de valoare	<code>sort()</code>	0 => PHP 1 => Perl 2 => Python
Sortarea unui tablou asociativ în funcție de valoare	<code>asort()</code>	20 => PHP 10 => Perl 21 => Python
Sortarea unui tablou după valoare, în ordine descendentă	<code>rsort()</code>	0 => Python 1 => Perl 2 => PHP
Sortarea unui tablou asociativ după valoare, în ordine descendentă	<code>Arsort()</code>	21 => Python 10 => Perl 20 => PHP
Sortarea unui tablou sau a unui tablou asociativ în funcție de cheie	<code>ksort()</code>	10 => Perl 20 => PHP 21 => Python
Sortarea unui tablou sau a unui tablou asociativ în funcție de cheie, în ordine descendentă	<code>Krsort()</code>	21 => Python 20 => PHP 10 => Perl



### Sfatul specialistului

**Întrebare:** Să presupunem că doresc să execut o căutare fără sensibilitate la diferența între majuscule și minuscule. Cum pot proceda?

**Răspuns:** O modalitate ar fi utilizarea funcției `natcasesort()`, care sortează un tablou folosind o „ordine naturală”, care nu este sensibilă la diferența între majuscule și minuscule. O altă modalitate constă în a utiliza funcția `usort()` sau una dintre funcțiile sale conexe sale, în speță `uksort()` și `uasort()`. Aceste funcții vă permit să definiți o secvență de aranjare personalizată, pe care o specificați prin desemnarea unei funcții care compară valorile în conformitate cu secvența de aranjare. Funcția `usort()` sortează valorile din tablou și returnează un tablou secvențial; funcția `uksort()` sortează cheile tabloului, iar funcția `uasort()` sortează un tablou asociativ.

De exemplu, următoarele instrucțiuni creează un tablou și îl sortează într-o manieră insensibilă la diferența între majuscule și minuscule:

```
$limbaje = array("Perl", "PHP", "Python");
usort($limbaje, "strncase");
```

Funcția `strncase()` este o funcție din biblioteca PHP care compară două șiruri fără a ține cont de mărimea literelor (majuscule sau minuscule). Funcția returnează o valoare negativă dacă primul șir este mai mic decât al doilea, zero dacă șirurile sunt identice, respectiv o valoare pozitivă dacă primul șir este mai mare decât al doilea.

Puteți implementa o secvență de aranjare personalizată scriind propria dumneavoastră funcție și specificând numele acesteia ca argument al funcției `usort()` sau al uneia din funcțiile sale conexe. Pur și simplu scrieți o funcție care preia două argumente șir și returnează -1, 0 sau +1, în funcție de rezultatul comparației între valorile șir.

Pentru mai multe informații despre funcția `usort()` și despre funcțiile sale conexe, examinați documentația în variantă electronică, la adresa <http://www.php.net>.



### Test „la minut”

- Care este funcția ce trebuie folosită pentru a sorta un tablou asociativ în ordine descrescătoare?
- Care este funcția ce trebuie folosită pentru a sorta un tablou în ordine ascendentă, în funcție de valoarea cheii?

Răspunsuri la test:

- `arsort()`
- `ksort()`



## Proiect 8-1: O reluare a formularului cu persoane de contact

În cadrul acestui proiect, veți vedea un exemplu simplu care include parcurgerea iterativă a tablourilor și sortarea acestora. Proiectul folosește sistemul de contact prin e-mail utilizat în proiectele anterioare. Formularul de contact prin e-mail furnizează spațiu pentru introducerea numelor copiilor persoanei de contact. Acest proiect prezintă modul de sortare a numelor în ordine alfabetică și modul de afișare a numelor respective.

### Scopurile proiectului

- Prezentarea modului de utilizare a unui tablou ca variabilă de formular
- Prezentarea modului de parcurgere iterativă a unui tablou
- Prezentarea modului de sortare a unui tablou

### Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit p-8-1.php și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
foreach ($copii as $index => $copil)
{
 echo "
copil[$index]=$copil";
}
echo "
";

sort($copii);

foreach ($copii as $index => $copil)
{
 echo "
copil[$index]=$copil";
}
?>
```

2. Plasați următorul text HTML într-un fișier denumit p-8-1.html și încărcați acest fișier în serverul dumneavoastră, plasându-l în același catalog ca și fișierul p-8-1.php:

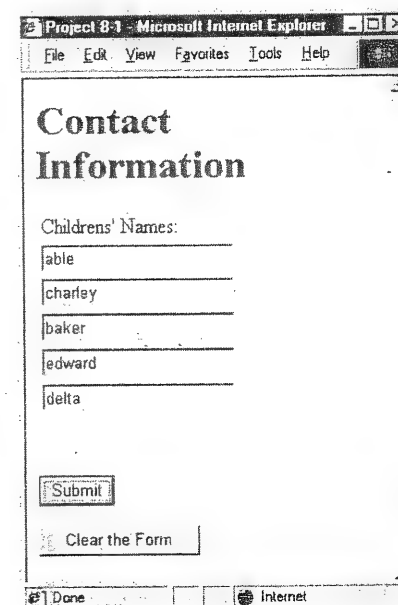
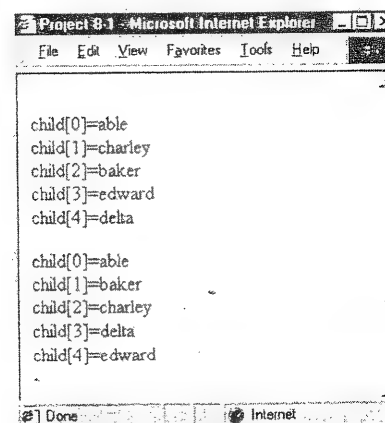
```
<HTML>
<HEAD>
<TITLE>Proiect 8-1</TITLE>
</HEAD>
<BODY>
<!-- Fișier p-8-1.html -->
<FORM METHOD="POST" ACTION="p-8-1.php">
<H1>Informații privind persoana de contact</H1>
<TABLE>
<TR>
<TD>Numele copiilor</TD>
</TR>
<TR>
<TD><INPUT TYPE="TEXT" NAME="copii[]"></TD>
</TR>
<TR>
<TD><INPUT TYPE="TEXT" NAME="copii[]"></TD>
</TR>
```

```
<TR>
<TD><INPUT TYPE="TEXT" NAME="copii[]"></TD>
</TR>
<TR>
<TD><INPUT TYPE="TEXT" NAME="copii[]"></TD>
</TR>
<TR>
<TD><INPUT TYPE="TEXT" NAME="copii[]"></TD>
</TR>
</TABLE>

<INPUT TYPE="SUBMIT" VALUE="Trimite">

<INPUT TYPE="RESET" VALUE="Sterge datele">
</FORM>
</BODY>
</HTML>
```

3. Alocăți un timp studiului paginii HTML. Casetele text sunt nume care fac referire la o variabilă de tip tablou PHP. De asemenea, parcurgeți scriptul PHP, studiind modul de funcționare a iterației care afișează numele copiilor.
4. Orientați un browser Web spre adresa URL a fișierului HTML încărcat anterior. Ecranul browserului trebuie să fie asemănător celui prezentat în continuare. Introduceți valori pentru numele copiilor și apoi executați clic pe butonul „Trimite”.
5. La execuția scriptului, acesta afișează numele copiilor în ordinea în care au fost introduse. Apoi, numele sunt sortate și afișate în ordine sortată. Un rezultat caracteristic al operației de sortare este prezentat în continuare.





### Test de evaluare

1. Scrieți instrucțiuni care creează un tablou denumit \$pop, care asociază numele mai multor orașe mari cu numărul locuitorilor acestora.
2. Scrieți o instrucțiune for care parcurge în mod iterativ un tablou secvențial denumit \$pitici, unde cheia minimă are valoarea unu. Corpul instrucțiunii for trebuie să afișeze numele fiecărui element al tabloului \$pitici. Aveți grijă la scrierea expresiei de test, care trebuie să reflecte faptul că valoarea cea mai mică a unei chei este unu, nu zero.
3. Scrieți o instrucțiune foreach care caută în tabloul \$state un element a cărui cheie are aceeași valoare ca și variabila \$abrev. Afișați valoarea elementului corespunzător, nu cheia acestuia.
4. Scrieți o instrucțiune care sortează tabloul asociativ \$pop în ordine crescătoare, în funcție de valoare.

## Modulul 9: Utilizarea șirurilor

### Scopuri

- **Învățați să utilizați secvențe escape suplimentare pentru a include în șiruri caracterele speciale**
- **Învățați să folosiți șiruri încadrate între ghilimele simple**
- **Învățați să utilizați codurile ASCII**
- **Învățați să creați date de ieșire formate**
- **Învățați să folosiți o varietate de funcții șir PHP, care vă permit să căutați și să manipulați șiruri**

În Modulul 2 au fost prezentate șirurile, tipul de date PHP care stochează texte. În cadrul modulului de față, veți învăța mai multe despre utilizarea șirurilor.

### Crearea și afișarea șirurilor

Până acum, ați învățat numeroase aspecte despre șiruri. Să recapitulăm noțiunile elementare înainte de a aborda subiecte mai complexe:

- Valorile literale de tip șir sunt secvențe de caractere incluse între ghilimele duble.
- Puteți include un caracter special în cadrul unui șir folosind o secvență escape care reprezintă caracterul special.
- Variabilele pot fi de tip string și li se poate atribui o valoare de tip șir prin intermediul operatorului de atribuire.
- Operatorul de concatenare (.) se poate folosi pentru unirea șirurilor.
- PHP poate converti în mod automat o valoare numerică într-un șir, respectiv valoarea unui șir într-un număr.

Dacă vreunul din aceste aspecte fundamentale vi se pare necunoscut, parcurgeți materialul corespunzător din capitolele 2 sau 5 înainte de a continua.

### Secvențe escape suplimentare

Modulul 2 a prezentat numeroase secvențe escape pe care le puteți folosi pentru a include caractere speciale în cadrul șirurilor. Cu toate acestea, PHP include două secvențe escape care nu au fost descrise în Modulul 2. Tabelul 9-1 prezintă setul complet de secvențe escape folosite în PHP.

Tabelul 9-1 Secvențele escape folosite în PHP

Secvența escape	Semnificație
<code>\n</code>	salt la linie nouă
<code>\r</code>	retur de car
<code>\t</code>	caracter de tabulare pe orizontală
<code>\\</code>	backslash
<code>\\$</code>	simbolul dolarului
<code>\"</code>	ghilimele duble
<code>\xxx</code>	caracterul asociat valorii ASCII xxx, exprimată sub forma unui număr în octal
<code>\xnn</code>	caracterul asociat valorii ASCII xxx, exprimată sub forma unui număr în hexazecimal

Ultimele două secvențe escape prezentate în tabelul 9-1 nu au fost descrise în Modulul 2. Fiecare dintre aceste secvențe escape folosește un cod ASCII (American Standard Code for Information Interchange) pentru reprezentarea unui caracter. Codurile ASCII sunt valori întregi, care sunt cuprinse între 0 și 255; fiecare literă sau simbol folosit frecvent în limbile vest-europene are asociat un cod ASCII. De exemplu, codul ASCII asociat literei A este 65, iar codul asociat cifrei 1 este 49.

Prima secvență escape necunoscută vă permite să folosiți reprezentarea în octal (baza 8) a unui cod ASCII pentru specificarea caracterului corespunzător. De exemplu, valoarea zecimală 65 (care este codul ASCII al literei A) poate fi reprezentată în octal sub forma 101. Ca atare, puteți reprezenta litera A folosind secvența escape `"\101"`.

Cea de-a doua secvență escape necunoscută vă permite să folosiți reprezentarea hexazecimală (în baza 16) a unui cod ASCII pentru specificarea caracterului corespunzător. De exemplu, valoarea zecimală 65 (care este codul ASCII al literei A) poate fi reprezentat sub forma hexazecimală 41. Ca atare, puteți reprezenta litera A cu ajutorul secvenței escape `„\41“`.

Dacă aveți nevoie de o reactualizare a cunoștințelor dumneavoastră în materie de lucrul cu baze de numerație diferite, să ne reamintim că valoarea unui număr zecimal este suma produselor dintre fiecare cifră care îl compune și o putere a lui 10. De exemplu, numărul zecimal 123 are valoarea  $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 = 1 \times 100 + 2 \times 10 + 3 \times 1$ . Puteți determina valoarea unui număr reprezentat folosind o altă bază prin înlocuirea lui 10 cu valoarea bazei respective într-o expresie ca aceea prezentată anterior. De exemplu, valoarea hexazecimală 123 este echivalentă cu valoarea zecimală  $1 \times 16^2 + 2 \times 16^1 + 3 \times 16^0 = 1 \times 256 + 2 \times 16 + 3 \times 1 = 291$ . Similar, valoarea în octal 123 este echivalentă cu valoarea zecimală  $1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 = 1 \times 64 + 2 \times 8 + 3 \times 1 = 83$ . Dacă algebra nu a fost materia dumneavoastră preferată, nu intrați în panică; în sub-sețiunea următoare veți învăța să determinați valoarea în octal sau hexazecimal echivalentă cu orice valoare zecimală.

## Utilizarea codurilor ASCII

Numeroase funcții PHP sunt deosebit de folositoare la utilizarea codurilor ASCII. Aceste funcții sunt enumerate în tabelul 9-2.

Iată un scurt exemplu de utilizare a funcției `ord()` pentru a determina codul ASCII corespunzător unui anumit caracter:

```
$c = "A";
$n = ord($c);
echo "
Valoarea ASCII a caracterului $c este $n";
```

Datele de ieșire ale exemplului respectiv sunt următoarele:

Valoarea ASCII a caracterului A este 65

Iată un exemplu care afișează echivalentele în zecimal, octal, respectiv hexazecimal al caracterelor ASCII ale căror coduri sunt cuprinse între 32 și 127:

```
for ($i=32; $i<128; $i++)
{
 $c = chr($i);
 $octal = decoct($i);
 $hex = dechex($i);
 echo "
$i (octal $octal, hex $hex): $c";
}
```

Domeniul dat include caracterele ASCII care se pot afișa. Iată o selecție din datele de ieșire ale exemplului:

```
32 (octal 40, hex 20):
33 (octal 41, hex 21):!
34 (octal 42, hex 22):"
35 (octal 43, hex 23):#
36 (octal 44, hex 24):$
```

Remarcați corespondența codului ASCII 32 cu un caracter spațiu.

Tabelul 9-2 Funcții folositoare în utilizarea codurilor ASCII

Funcție	Descriere
<code>chr(<i>n</i>)</code>	Returnează caracterul având codul ASCII dat de <i>n</i> .
<code>dechex(<i>n</i>)</code>	Returnează valoarea hexazecimală echivalentă cu valoarea zecimală dată de <i>n</i> .
<code>decoct(<i>n</i>)</code>	Returnează valoarea în octal echivalentă cu valoarea zecimală dată de <i>n</i> .
<code>hexdec(<i>n</i>)</code>	Returnează valoarea zecimală echivalentă cu valoarea hexazecimală dată de <i>n</i> .
<code>octdec(<i>n</i>)</code>	Returnează valoarea zecimală echivalentă cu valoarea în octal dată de <i>n</i> .
<code>ord(<i>c</i>)</code>	Returnează codul ASCII echivalent caracterului <i>c</i> .

## Șiruri delimitate între ghilimele simple

Dacă preferați, puteți delimita un șir între ghilimele simple, nu neapărat duble. Un motiv în acest sens îl constituie facilitarea posibilității de includere a ghilimelelor



duble în șir; dacă delimitați un șir între ghilimele duble, toate ghilimelele duble din cadrul șirului trebuie corelate cu un caracter backslash:

```
"\"Stop\", striga el, \"sau trag!\""
```

Acesta este un șir care conține nume-roase ghilimele duble. – N.T.

Acest șir poate fi scris într-un mod mai convenabil astfel:

```
"Stop", striga el, "sau trag!"
```

Șirurile delimitate între ghilimele simple se comportă într-un mod diferit față de șirurile delimitate prin ghilimele duble:

- Singurele secvențe escape permise în cadrul șirurilor delimitate prin ghilimele simple sunt `\\` și `\'`.
- Nu se execută substituția variabilelor atunci când datele de ieșire sunt reprezentate printr-un șir delimitat prin ghilimele simple.

În consecință, datele de ieșire ale următorului exemplu

```
$x = "test";
echo "x este $x";
```

sunt următoarele:

```
x este $x
```

și nu

```
x este test
```

## Crearea datelor de ieșire formate

PHP include două funcții utile pentru generarea datelor de ieșire formate, în speță `printf()` și `sprintf()`. Funcția `printf()` afișează datele sale de ieșire, în timp ce funcția `sprintf()` returnează datele sale de ieșire sub forma unei valori șir. În general, fiecare funcție precia două sau mai multe argumente. Primul argument este un șir, denumit *șir de formatare*, care specifică formatul datelor de ieșire, iar celelalte argumente specifică valorile care vor constitui datele de ieșire. Iată un exemplu simplu de utilizare a funcției `printf()`:

```
printf("Valoarea lui n este: %d", $n);
```

Iată și un exemplu mai complicat, care prezintă modul de utilizare a funcției `sprintf()` cu mai multe argumente pentru datele de ieșire:

```
$rezultat = sprintf("Valori: %d, %f", $n, $x);
```

Șirul de formatare constă dintr-o serie de caractere și directive ordinare. Un caracter ordinar este orice caracter, în afara caracterului `%`. Caracterele ordinare sunt pur și simplu copiate la ieșire.

Directivele reprezintă secvențe de caractere care încep cu simbolul `%`; aceste determină modul în care va fi formatat argumentul corespunzător. O directivă simplă, cum este cea din exemplul precedent, poate consta dintr-un caracter `%` urmat de un specificator de tip, precum `d`, care arată că argumentul trebuie tratat ca număr zecimal. Cu toate acestea, o directivă mai sofisticată poate include următoarele componente, care trebuie să apară în ordinea indicată:

- **Specificator de completare** (opțional): Un specificator de completare precizează caracterul care se va folosi pentru a completa rezultatul până la dimensiunea cerută a șirului. În cazul în care caracterul de completare este omis, rezultatul este completat cu spații. Specificatorul de completare poate fi un caracter spațiu sau un `0` (zero). Un specificator de completare de tip spațiu este folosit frecvent cu șirurile, iar un specificator de completare zero se folosește mai ales alături de numere. Un alt caracter de completare poate fi specificat prin prefixarea acestuia cu un singur semn al citării (`'`). De exemplu, pentru a completa un rezultat cu liniuțe de subliniere, specificați `'_'` drept caracter de completare.
- **Specificator de aliniere** (opțional): Un specificator de aliniere indică dacă rezultatul trebuie să fie aliniat la stânga sau la dreapta. Dacă specificatorul de aliniere este omis, rezultatul va fi aliniat la dreapta; dacă se indică o cratimă (`-`) drept specificator de aliniere, rezultatul va fi aliniat la stânga.
- **Specificator de lățime** (opțional): Un specificator de lățime este un întreg care determină numărul minim de caractere ale rezultatului; sau, dacă argumentul este de tip `double`, numărul minim de caractere situate la stânga punctului zecimal. Dacă rezultatul conține un număr mai redus de caractere, atunci va conține și caractere de completare.
- **Specificator de precizie** (opțional): Un specificator de precizie este un punct zecimal, urmat de un întreg care determină numărul de cifre după punctul zecimal pe care trebuie să le conțină rezultatul. Specificatorul de precizie nu are nici un efect pentru alte tipuri decât `double`.
- **Specificator de tip** (obligatoriu): Specificatorul de tip determină modul de tratare și afișare a argumentului. Tabelul 9-3 rezumă specificatorii de tip disponibili.

**Tabelul 9-3** Specificatorii de tip PHP folosiți la formatarea șirurilor

Specificator de tip	Descriere
<code>b</code>	Tratează argumentul ca pe un întreg și îl afișează ca valoare binară.
<code>c</code>	Tratează argumentul ca pe un întreg și afișează caracterul cu aceeași valoare ASCII ca și argumentul.
<code>d</code>	Tratează argumentul ca pe un întreg și îl afișează ca valoare zecimală.
<code>f</code>	Tratează argumentul ca pe o valoare de tip <code>double</code> și îl afișează ca valoare cu virgulă mobilă.

continuare

Tabelul 9-3 Continuare

Specificator de tip	Descriere
<code>o</code>	Tratează argumentul ca pe un întreg și îl afișează ca pe o valoare scrisă în octal.
<code>s</code>	Tratează argumentul ca pe un șir și îl afișează.
<code>x</code>	Tratează argumentul ca pe un întreg și îl afișează ca număr hexazecimal, cu litere scrise cu minuscule.
<code>X</code>	Tratează argumentul ca pe un întreg și îl afișează ca număr hexazecimal, cu litere scrise cu majuscule.

Dacă doriți să inserați simbolul procentului în datele de ieșire ale unui apel la funcția `printf()` sau `sprintf()`, inserați două caractere `%` în locul dorit. La apariția a două caractere `%`, funcțiile `printf()` și `sprintf()` știu că doriți să inserați un simbol al procentului, nu o directivă de formatare.

Tabelul 9-4 prezintă rezultatele aplicării a diferite șiruri de formatare valorilor selectate. Studiați tabelul și verificați dacă ați înțeles corect modul de operare a specificatorilor din cadrul fiecărui șir de formatare. Remarcați că, în cazul omiterii cifrelor zecimale, se produce automat o rotunjire.

### Notă

În tabelul 9-4, spațiile au fost înlocuite prin accente circumflexe, pentru a facilita determinarea numărului de spații și a amplasării acestora.

Tabelul 9-4 Exemple de rezultate ale utilizării a diferite șiruri de formatare

Valoare	Format	Rezultat
100	<code>,"%d"</code>	100
100	<code>,"%b"</code>	1100100
100	<code>,"%o"</code>	144
100	<code>,"%x"</code>	64
100	<code>,"%f"</code>	100.000000
12.345	<code>,"%'^-10f"</code>	12.345000 ^ ^ ^ ^ ^ ^
12.345	<code>,"%'^10f"</code>	^ ^ ^ ^ ^ ^ 12.345000
12.345	<code>,"%'^-10.2f"</code>	12.35 ^ ^ ^ ^ ^ ^
12.345	<code>,"%'10.2f"</code>	^ ^ ^ ^ ^ ^ 12.35
„test”	<code>,"%'^-10s"</code>	test ^ ^ ^ ^ ^ ^
„test”	<code>,"%'^10s"</code>	^ ^ ^ ^ ^ ^ test



### Sfatul specialistului

**Întrebare:** Utilizarea funcțiilor `printf()` și `sprintf()` pentru formatarea numerelor pare cam greoaie. Nu există și alte metode?

**Răspuns:** Dacă preferați, puteți folosi funcția `number_format()`, care returnează o valoare de tip șir conținând un rezultat formatat. Puteți apela funcția cu unul, două sau patru argumente:

```
number_format(numar)
number_format(numar, zecimale)
number_format(numar, zecimale, punct_zecimal, separator_mii)
```

Argumentul *numar* specifică valoarea numerică pe care doriți să o formatați. Argumentul *zecimale* specifică numărul dorit de cifre zecimale. Argumentul *punct\_zecimal* precizează caracterul ce se va folosi drept punct zecimal, iar argumentul *separator\_mii* precizează caracterul care se va folosi ca separator al miilor. În mod prestabilit, rezultatul este formatat fără zecimale, este inserat un punct (.) înaintea cifrelor care compun partea zecimală, respectiv se folosește o virgulă (,) pentru separarea miilor.

De exemplu, apelul la funcția

```
number_format(1.234, 2)
```

returnează valoarea

1.23



### Test „la minut”

- Scrieți secvența escape care reprezintă caracterul a cărui valoare ASCII este dată de valoarea în octal 17.
- Scrieți secvența escape care reprezintă caracterul a cărui valoare ASCII este dată de valoarea hexazecimală 1f.
- Denumiți funcția care returnează valoarea hexazecimală a unui număr zecimal.
- Scrieți un șir de formatare care generează un număr cu virgulă mobilă cu două cifre zecimale.

Răspunsuri la test:

- `"\017"`
- `"\x1f"`
- `dechex()`
- `"%.2f"`

## Manipularea șirurilor

PHP conține peste 70 de funcții care lucrează cu șiruri. Această secțiune descrie numeroase funcții pe care este posibil să le utilizați frecvent. Aceste funcții vă permit să obțineți lungimea unui șir, să eliminați dintr-un șir caracterele de tip spațiu alb și să converțiți caracterele unui șir în majuscule sau minuscule.

### Obținerea lungimii unui șir

Funcția `strlen()` returnează lungimea șirului specificat ca argument al funcției. Iată un exemplu simplu de utilizare a funcției `strlen()`:

```
$s = "Acesta este un șir.";
$n = strlen($s);
echo "
Lungimea șirului este: $n";
```

Acesta este rezultatul generat de exemplul de mai sus:

```
Lungimea șirului este: 19
```

### Eliminarea caracterelor dintr-un șir

Numeroase funcții PHP vă permit să eliminați caracterele de tip spațiu alb de la una sau ambele extremități ale unui șir. Caracterele de tip spațiu alb sunt caractere precum spațiu, tabulator și caracter de salt la linie nouă, care nu dispun de nici o reprezentare vizibilă. Aceste funcții sunt prezentate în tabelul 9-5. Iată un exemplu care prezintă modul de operare al acestor funcții.

```
$s = " Acesta este un șir. ";
$n = strlen($s);
echo "
Lungimea șirului este: $n";

$trim = chop($s);
$n = strlen($trim);
echo "
Lungimea șirului este: $n";

$trim = ltrim($s);
$n = strlen($trim);
echo "
Lungimea șirului este: $n";

$trim = rtrim($s);
$n = strlen($trim);
echo "
Lungimea șirului este: $n";

$trim = trim($s);
$n = strlen($trim);
echo "
Lungimea șirului este: $n";
```

Iată și datele de ieșire ale exemplului:

```
Lungimea șirului este: 39
Lungimea șirului este: 29
Lungimea șirului este: 29
Lungimea șirului este: 29
Lungimea șirului este: 19
```

**Tabelul 9-5** Funcții PHP de eliminare a caracterelor din șiruri

Funcție	Descriere
<code>chop(s)</code>	Returnează valoarea lui <code>s</code> , eliminând spațiile albe de la extremitatea din dreapta a șirului. Similar cu <code>rtrim()</code> .
<code>ltrim(s)</code>	Returnează valoarea lui <code>s</code> , eliminând spațiile albe de la extremitatea din stânga a șirului.
<code>rtrim(s)</code>	Returnează valoarea lui <code>s</code> , eliminând spațiile albe de la extremitatea din dreapta a șirului. Similar cu <code>chop()</code> .
<code>trim(s)</code>	Returnează valoarea lui <code>s</code> , eliminând spațiile albe de la ambele extremități.

### Conversia șirurilor la majuscule sau minuscule

Funcția `strtoupper()` returnează valoarea argumentului său, convertită la majuscule. Funcția conexă `strtolower()` returnează valoarea argumentului său, convertită la minuscule. Nici una din funcții nu modifică valoarea argumentului său; valoarea convertită este cea returnată ca rezultat al funcției. Iată un scurt exemplu, care prezintă modul de operare a acestor funcții:

```
$s = "abcd";
$rezultat = strtoupper($s);
echo "
strtoupper($s): $rezultat";
$s = "ABCD";
$rezultat = strtolower($s);
echo "
strtolower($s): $rezultat";
```

Iată și datele de ieșire generate de exemplul respectiv:

```
strtoupper(abcd): ABCD
strtolower(ABCD): abcd
```



#### Test „la minut”

- Care este funcția ce returnează lungimea unui șir?
- Care este funcția care elimină caracterele de tip spațiu alb de la ambele extremități ale unui șir?
- Care este funcția care convertește caracterele unui șir în minuscule?

Răspunsuri la test:

- `strlen()`
- `trim()`
- `strtolower()`

## ? Sfatul specialistului

**Întrebare:** Ați spus că PHP include peste 70 de funcții care utilizează șiruri, dar ați descris numai câteva. Cum pot afla detalii și despre celelalte funcții?

**Răspuns:** Secțiunea următoare descrie unele funcții PHP și care compară șiruri și execută căutări în șiruri. Cu toate acestea, PHP include mult mai multe funcții și decât cele care pot fi descrise în acest capitol. Pentru a afla mai multe detalii despre funcțiile și suplimentare, consultați manualul PHP pe suport electronic, la adresa <http://www.php.net>.

## Compararea șirurilor și căutarea în șiruri

În secțiunea anterioară ați învățat despre numeroase funcții PHP care manipulează șiruri. În această secțiune, veți afla detalii despre numeroase funcții PHP care execută anumite categorii de operații cu șiruri, și anume compararea și căutarea.

### Compararea șirurilor

- PHP furnizează patru funcții care sunt deosebit de utile pentru compararea șirurilor. Aceste funcții sunt enumerate în tabelul 9-6. Fiecare funcție returnează o valoare al cărei semn determină rezultatul comparației; nu trebuie să încercați să interpretați valoarea returnată efectivă. Funcția `strncasecmp()` a fost adăugată în versiunea PHP 4.0.2; dacă dispuneți de o versiune anterioară a limbajului PHP, nu aveți la dispoziție această funcție.

**Tabelul 9-6** Funcții PHP de comparație între șiruri

Funcție	Descriere
<code>strcmp(s1, s2)</code>	Execută o comparație fără sensibilitate la diferența între majuscule și minuscule. Returnează o valoare mai mică decât zero dacă <code>s1</code> este mai mic decât <code>s2</code> , o valoare mai mare decât zero dacă <code>s1</code> este mai mare decât <code>s2</code> , respectiv 0 în celelalte cazuri.
<code>strcmp(s1, s2)</code>	Execută o comparație cu sensibilitate la diferența între majuscule și minuscule. Returnează o valoare mai mică decât zero dacă <code>s1</code> este mai mic decât <code>s2</code> , o valoare mai mare decât zero dacă <code>s1</code> este mai mare decât <code>s2</code> , respectiv 0 în celelalte cazuri.
<code>strncasecmp(s1, s2, n)</code>	Execută o comparație fără sensibilitate la diferența între majuscule și minuscule. Returnează o valoare mai mică decât zero dacă <code>s1</code> este mai mic decât <code>s2</code> , o valoare mai mare decât zero dacă <code>s1</code> este mai mare decât <code>s2</code> , respectiv 0 în celelalte cazuri. La comparație sunt luate în considerare un număr de <code>n</code> caractere.
<code>strncmp(s1, s2, n)</code>	Execută o comparație cu sensibilitate la diferența între majuscule și minuscule. Returnează o valoare mai mică decât zero dacă <code>s1</code>

### Funcție

### Descriere

este mai mic decât `s2`, o valoare mai mare decât zero dacă `s1` este mai mare decât `s2`, respectiv 0 în celelalte cazuri. La comparație sunt luate în considerare un număr de `n` caractere.

Iată un scurt exemplu care prezintă modul de utilizare a acestor funcții, precum și datele de ieșire ale exemplului:

```
$s1 = "abcd";
$s2 = "ABCE";

$rezultat = strcmp($s1, $s2);
echo "
Funcția strcmp a returnat $rezultat.";

$rezultat = strcmp($s1, $s2);
echo "
Funcția strcmp a returnat $rezultat.";

//PHP 4.0.2+
$rezultat = strncasecmp($s1, $s2);
echo "
Funcția strncasecmp a returnat $rezultat.";

$rezultat = strncmp($s1, $s2, 3);
echo "
Funcția strncmp a returnat $rezultat.";
```

Iată și datele de ieșire ale exemplului:

```
Funcția strcmp a returnat 1.
Funcția strncmp a returnat 1.
Funcția strncasecmp a returnat -1.
Funcția strncasecmp a returnat 0.
```

Datele de ieșire vă arată că funcția `strcmp()` a identificat șirul "abcd" ca fiind mai mare decât "ABCE", ca de altfel și funcția `strncmp()`. Aceasta s-a întâmplat deoarece literele minuscule au în secvența ASCII o poziție superioară literelor scrise cu majuscule; litera A are valoarea ASCII 65, iar litera a are valoarea ASCII 97. De asemenea, datele de ieșire arată că funcția `strncasecmp()` a identificat șirul "abcd" ca fiind mai mic decât "ABCE", precum și că funcția `strncasecmp()` a identificat șirul "abcd" ca fiind egal cu "ABCE".

## Descoperirea și extragerea sub-șirurilor

PHP include numeroase funcții care găsesc și extrag *sub-șiruri*, adică părți dintr-un șir. Cele mai importante funcții de acest gen sunt rezumate în tabelul 9-7.

**Tabelul 9-7** Funcții PHP de extragere și căutare

Funcție	Descriere
<code>strchr(s1, s2)</code>	Returnează toate șirurile <code>s1</code> de la prima apariție a șirului <code>s2</code> și până la sfârșit. Dacă <code>s1</code> nu este găsit, funcția returnează false. Funcția <code>strstr()</code> execută aceeași operație.

continuare

Tabelul 9-7. Continuare

Funcție	Descriere
<code>stristr(s1,s2)</code>	Returnează toate șirurile <i>s1</i> de la prima apariție a șirului <i>s2</i> și până la sfârșit. Dacă <i>s1</i> nu este găsit, funcția returnează false. Șirurile <i>s1</i> și <i>s2</i> sunt comparate fără a se ține cont dacă literele sunt majuscule sau minuscule.
<code>strpos(s1,s2)</code>	Returnează poziția întreagă a primei apariții a șirului <i>s2</i> în <i>s1</i> . Dacă <i>s2</i> nu este găsit, funcția returnează false.
<code>strrchr(s1,s2)</code>	Returnează toate șirurile <i>s1</i> de la ultima apariție a șirului <i>s2</i> și până la sfârșit. Dacă <i>s1</i> nu este găsit, funcția returnează false. La comparație este folosit numai primul caracter al șirului <i>s2</i> .
<code>strstr(s1,s2)</code>	Returnează toate șirurile <i>s1</i> de la prima apariție a șirului <i>s2</i> și până la sfârșit. Dacă <i>s1</i> nu este găsit, funcția returnează false. Funcția <code>strchr()</code> execută aceeași operație.
<code>substr(s,start)</code> <code>substr(s,start,lung)</code>	Returnează porțiunea șirului <i>s</i> specificată de indexul întreg <i>start</i> , respectiv de indexurile <i>start</i> și <i>lung</i> . Prima poziție a șirului este poziția 0.

Iată un exemplu simplu, care prezintă modalitățile de utilizare a mai multor funcții de extragere și căutare:

```
$s = "the cat on the mat near the bat";
$b = "at";
$n = strpos($s, $b);
echo "
strpos(\"$s\", \"$b\"): $n";

$s = "the cat on the mat near the bat";
$b = "at";
$n = strchr($s, $b);
echo "
strchr(\"$s\", \"$b\"): $r";

$s = "the cat on the mat near the bat";
$b = "at";
$n = strrchr($s, $b);
echo "
strrchr(\"$s\", \"$b\"): $r";

$s = "the cat on the mat near the bat";
$rezultat = substr($s, 4, 3);
echo "
substr(\"$s\", 4, 3): $rezultat";
```

Iată și datele de ieșire ale exemplului:

```
strpos("the cat on the mat near the bat", "at"): 5
strchr("the cat on the mat near the bat", "at"): at on the mat near the bat
strrchr("the cat on the mat near the bat", "at"): at
substr("the cat on the mat near the bat", 4, 3): cat
```

O potențială dificultate în utilizarea funcției `strpos()` constă în aceea că poate fi greu de sesizat diferența dintre valoarea returnată 0, care arată că sub-șirul a fost găsit în poziția inițială a șirului, și valoarea returnată false, care arată că sub-șirul nu a fost găsit. Iată un scurt exemplu care indică un mod adecvat de testare a valorii

returnate de funcția `strpos()`, astfel încât să se poată face diferența între cele două rezultate:

```
$poz = strpos($s1, $s2);
if ($poz === false)
{
 //sub-șirul nu a fost găsit...
}
```

Procedeu prezentat folosește operatorul de identitate (`===`) pentru a determina dacă valoarea returnată este identică – nu doar aritmetic egală – cu valoarea false. Dacă folosiți în schimb operatorul de egalitate, este posibil ca rezultatul să fie incorect; false are valoarea numerică zero, valoare returnată și dacă sub-șirul este găsit în poziția inițială a șirului.

### Avertisment

Acest procedeu presupune utilizarea versiunii PHP 4.0b3 sau a unei versiuni ulterioare; în versiunile anterioare ale limbajului PHP, funcția `strpos()` returna o valoare șir. Dacă folosiți o versiune anterioară a limbajului PHP, examinați manualul PHP pentru mai multe informații referitoare la funcția `strpos()`.

## Înlocuirea unui sub-șir

O operație frecvent folosită în programare constă în găsirea unui sub-șir și înlocuirea sa cu o valoare nouă. PHP are două funcții deosebit de utile pentru asemenea operații, și anume `str_replace()` și `substr_replace()`. În tabelul 9-8 sunt prezentate pe scurt aceste funcții. Remarcați că funcția `str_replace()` notează sub-șirul prin valoarea sa, în timp ce funcția `substr_replace()` notează sub-șirul prin poziția sa în interiorul șirului subiect.

Tabelul 9-8. Funcții PHP de înlocuire a sub-șirurilor

Funcție	Descriere
<code>str_replace(cauta, inlocuire, subiect)</code>	Se caută în șirul subiect sub-șirul <i>cauta</i> ; dacă sub-șirul este găsit, returnează valoarea <i>subiect</i> , înlocuindu-se prima apariție a șirului <i>cauta</i> cu <i>inlocuire</i> .
<code>substr_replace(subiect, inlocuire, start, lungime)</code>	Returnează valoarea <i>subiect</i> , înlocuind sub-șirul care începe de la <i>start</i> și având lungimea <i>lungime</i> cu șirul <i>inlocuire</i> .

Iată un exemplu care prezintă modul de utilizare a acestor funcții:

```
$subiect = "the cat on the mat near the bat";
$cauta = "cat";
$inlocuire = "CAT";
$rezultat = str_replace($cauta, $inlocuire, $subiect);
```

```
echo "
str_replace(\"$scauta\", \"$inlocuire\", \"$subiect\"): $rezultat";

$inlocuire = "CAT";
$rezultat = substr_replace($subiect, $inlocuire, 4, 3);
echo "
substr_replace(\"$subiect\", \"$inlocuire\", 4, 3): $rezultat";
```

Iată și datele de ieșire ale exemplului:

```
str_replace("cat", "CAT", "the cat on the mat near the bat"): the CAT on the
mat near the bat
substr_replace("the cat on the mat near the bat", "CAT", 4, 3): the CAT on the
mat near the bat
```

## Stabilirea unei identități între caractere

Una dintre cele mai puternice caracteristici ale limbajului PHP o constituie capacitatea acestuia de a folosi *expresiile regulate*, o sintaxă specială pentru specificarea seturilor de șiruri. Utilitatea cea mai frecventă a unei expresii regulate constă în a determina dacă un șir subiect conține sau nu un sub-șir identic cu expresia regulată respectivă. Veți învăța să efectuați operația respectivă mai târziu, pe parcursul sub-sectiunii. Înainte de a ajunge la aceasta, să învățăm să formăm expresii regulate.

### Scrierea expresiilor regulate

Să presupunem, de exemplu, că doriți să specificați un șir care poate include litera *b* sau litera *c*. Puteți face aceasta folosind expresia regulată `[bc]`. Prin includerea valorilor posibile între paranteze, formați o expresie regulată echivalentă cu formularea „alege oricare din aceste valori”.

Să presupunem că doriți să specificați un șir care poate include orice vocală; expresia regulată `[aeiou]` vă poate fi de ajutor. Dacă doriți să permiteți și utilizarea majusculilor, puteți scrie `[aeiouAEIOU]`.

Să presupunem că doriți să specificați un șir care poate include orice caracter scris cu minuscule. Puteți scrie:

```
[abcdefghijklmnopqrstuvwxyz]
```

Sau puteți folosi forma mai compactă `[a-z]`, unde prin cratimă se înțelege o serie de caractere consecutive.

Să presupunem că doriți să specificați șirurile "sat", "mat" și "lat". Pentru aceasta, aveți nevoie de expresia regulată `[sm]at`. Semnificația acestei expresii regulate este următoarea: „alege oricare din literele *s*, *m* și *l* și scrie după litera respectivă literele *at*”.

Dacă un accent circumflex este primul simbol menționat între parantezele drepte, acesta are ca efect inversarea semnificației expresiei regulate plasate între paranteze. De exemplu, expresia regulată `[^a-z]` corespunde oricărui caracter diferit de un caracter scris cu minuscule.

Pentru a specifica faptul că o expresie regulată se poate repeta, expresia regulată va fi urmată de o pereche de paranteze acolade, care includ limitele superioară și

inferioară ale repetiției. De exemplu, expresia regulată `[aeiou]{1,4}` corespunde unui șir care este compus din 1-4 vocale. Pentru a specifica repetarea mai multor părți ale unei expresii regulate, includeți părțile respective între paranteze. De exemplu, expresia regulată `([sm]at){1,2}` corespunde unui număr de una sau două repetări ale oricăruia dintre șirurile "sat", "mat" sau "lat".

Unele valori care se repetă sunt atât de frecvent folosite, încât au abrevieri:

Abreviere	Semnificație
+	{1, n}, unde n este un număr arbitrar de mare
*	{0, n}, unde n este un număr arbitrar de mare
?	{0,1}

Să presupunem că doriți să reprezentați o înmulțire. Dacă folosiți caractere minuscule pentru operanzi, puteți obține ceva de genul `[a-z]*[a-z]`. Totuși, această expresie regulată nu are semnificația dorită, deoarece `*` este un factor de repetiție, nu un caracter dintr-un șir. Pentru a dezactiva semnificația specială a caracterului `*`, trebuie să-l prefixați cu un caracter backslash: `[a-z]\*[a-z]`.

Pentru a specifica faptul că o expresie regulată corespunde numai unui sub-șir care include caracterul inițial al unui șir subiect, prefixați expresia regulată cu un accent circumflex. De exemplu, expresia regulată `^[sm]at` corespunde sub-șirurilor "sat", "mat" sau "lat" numai dacă acestea apar la începutul șirului subiect. Similar, pentru a arăta că o expresie regulată corespunde numai unui sub-șir care include caracterul final al unui șir subiect, anexați la expresia regulată un simbol al dolarului. De exemplu, expresia regulată `[sm]at$` corespunde șirurilor "sat", "mat" sau "lat" numai dacă acestea apar la sfârșitul șirului subiect. Expresia regulată `^[sm]at$` corespunde sub-șirurilor "sat", "mat" sau "lat" numai dacă șirul subiect este identic cu unul dintre aceste sub-șiruri.

### Sugestie

Expresiile regulate includ și alte instrumente în afara celor descrise până acum, instrumente al căror număr este prea mare pentru a putea fi explicate într-un singur capitol. Nici măcar în manualul PHP pe suport electronic nu există o descriere completă a expresiilor regulate. Dacă aveți acces la paginile de manual instalate cu PHP, puteți citi o descriere completă a expresiilor regulate acceptate de PHP, care sunt compatibile cu standardul POSIX 1003.2. Standardul POSIX este un document costisitor, protejat prin legislația drepturilor de autor. Dacă nu puteți obține accesul la paginile de manual PHP sau la standardul POSIX, puteți găsi un îndrumar util privind expresiile regulate, scris de Dario F. Gomes, la adresa <http://www.phpbuilder.com>.

### Utilizarea expresiilor regulate

PHP include numeroase funcții care lucrează cu expresii regulate. Tabelul 9-9 descrie aceste funcții. Sub-sectiunea de față explică funcția `ereg()`; pentru informații privind celelalte funcții, consultați manualul PHP.



Forma simplă a funcției `ereg()` preia două argumente: un șir care conține o expresie regulată și un șir subiect. Funcția returnează `true` dacă expresia regulată corespunde unui sub-șir al șirului subiect; în caz contrar, returnează `false`. Iată un exemplu simplu:

```
$model = "[sm]at";
$subject = "La noi în sat";
$resultat = ereg($model, $subject);
```

Variabila `$resultat` primește valoarea `true`, deoarece șirul subiect conține sub-șirul "sat", care corespunde expresiei regulate.

Forma mai complexă a funcției `ereg()` include un al treilea argument, un tablou care primește sub-șiruri ce corespund porțiunilor scrise între paranteze ale modelului. Proiectul aferent capitolului curent prezintă modul de utilizare a acestei forme a funcției `ereg()`, pe care o folosește pentru a determina dacă un șir conține o adresă de e-mail corect formată.

**Tabelul 9-9** Funcții PHP pentru expresii regulate

Funcție	Descriere
<code>ereg</code>	Execută o identificare cu o expresie regulată
<code>ereg_replace</code>	Înlocuiește un sub-șir care corespunde unei expresii regulate
<code>eregi</code>	Execută o identificare cu o expresie regulată insensibilă la diferența între majuscule și minuscule
<code>eregi_replace</code>	Înlocuiește un sub-șir care corespunde unei expresii regulate insensibile la diferența între majuscule și minuscule
<code>split</code>	Divide un șir într-un tablou folosind o expresie regulată
<code>spliti</code>	Divide un șir într-un tablou folosind o expresie regulată (insensibilă la diferența între majuscule și minuscule)
<code>sql_regcase</code>	Creează o expresie regulată insensibilă la diferența între majuscule și minuscule dintr-un șir care conține o expresie regulată.



### Sfatul specialistului

**Întrebare:** Acest capitol explică modul de generare a unor date de ieșire formate. Există vreo modalitate simplă de baleiere a datelor de intrare formate?

**Răspuns:** Funcția PHP `sscanf()`, adăugată în versiunea PHP 4.01, este complementara funcției `printf()`. Dacă funcția `printf()` generează date de ieșire formate, funcția `sscanf()` citește un șir, îl interpretează prin referirea la un șir de formare și stabilește valorile variabilelor specificate în funcție de conținutul șirului.

Să luăm în considerare următorul exemplu:

```
$subject = "01,31,2005";
$n = sscanf($subject, "%d,%d,%d", &$luna, &$zi, &$an);
```

```
echo "
Au fost gasite $n valori:";
echo "
luna=$luna";
echo "
zi=$zi";
echo "
an=$an";
```

Datele de ieșire ale acestui exemplu sunt:

```
Au fost gasite 3 valori:
luna=1
zi=31
an=2005
```

Pentru mai multe informații privind funcția `sscanf()`, consultați manualul PHP.



### Test „la minut”

- Ce funcție PHP execută căutarea sensibilă la diferența între majuscule și minuscule a unui șir subiect al unui sub-șir dat, examinând un număr maxim dat de caractere?
- Ce funcție PHP extrage un sub-șir al unui șir subiect, în funcție de poziția inițială și de lungimea sub-șirului?
- Scrieți o expresie regulată care corespunde numai sub-șirurilor "jos", "ros" și "cos" care apar la începutul unui șir.



### Proiect 9-1: O rutină de identificare a echivalențelor cu expresii regulate

În cadrul acestui proiect, veți construi o pagină care vă permite să introduceți un șir și o expresie regulată, precum și să căutați în șirul respectiv un sub-șir care corespunde expresiei regulate. Puteți folosi acest proiect pentru a vă îmbunătăți cunoștințele în materie de expresii regulate.

#### Scopurile proiectului

- Prezentarea modului de utilizare a funcției `ereg()`
- Prezentarea modului de testare a formularului cu date introduse de utilizator
- Prezentarea modului de validare a unei adrese de e-mail

#### Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit `p-9-1.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
echo "
șir: $șir";
echo "
expresie regulată: $model";
if (get_magic_quotes_gpc())
```

Răspunsuri la test:

- `strcmp()`
- `substr()`
- `^[jrc]os`

```

{
 echo "

";
 echo "
Eliminarea ghilimelelor magice....";
 $sir = stripslashes($sir);
 $model = stripslashes($model);
 echo "
sir: $sir";
 echo "
expresie regulata: $model";
}
$gasit = ereg($model, $sir, $echivalente);
echo "

";
if ($gasit)
{
 echo "
valabil: true";
 echo "

";
 echo "
Componente: ";
 for ($i = 0; $i < count($echivalente); $i++)
 {
 echo "
$echivalente[$i]";
 }
}
else
 echo "
valabil: false";
?>

```

2. Plasați următorul text HTML într-un fișier denumit p-9-1.html și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul p-9-1.php:

```

<HTML>
<HEAD>
<TITLE>Proiect 9-1</TITLE>
</HEAD>
<BODY>
<H2>Proiect 9-1: Testarea expresiilor regulate</H2>
<FORM METHOD="POST" ACTION="p-9-1.php">

Sir:

<INPUT TYPE="TEXT" NAME="sir">

Expresie regulata:

<INPUT TYPE="TEXT" SIZE=64 NAME="model">

VALUE="^([_a-zA-Z0-9-]+)@([_a-zA-Z0-9-]+)([a-zA-Z0-9-]+)*"$>

<INPUT TYPE="SUBMIT">

</FORM>
</BODY>
</HTML>

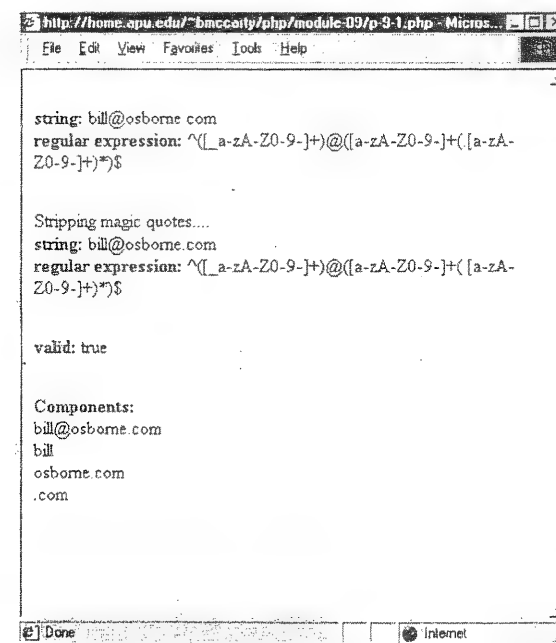
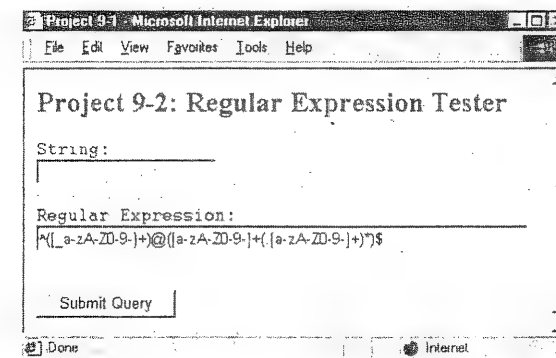
```

3. Alocați un timp studiului scriptului HTML. Observați apelul la funcția `get_magic_quotes_gpc()`. Această funcție returnează o valoare corespunzătoare opțiunii PHP `magic_quotes_gpc`, configurată de administratorul de sistem PHP. Dacă această opțiune este activată, PHP adaugă automat caractere backslash la variabilele din formular, astfel încât valorile lor să poată fi folosite în contexte care impun protecția caracterelor speciale. Această caracteristică nu este utilă la introducerea, vizualizarea și utilizarea unei expresii regulate.

Deci, dacă opțiunea este activată, scriptul folosește funcția `stripslashes()` pentru a elimina toate caracterele slash nedorite adăugate în mod automat de PHP. Pentru a vedea de ce este necesar acest lucru, puteți elimina instrucțiunea `if` și corpul său din script și apoi încercați să rulați scriptul. Dacă opțiunea `magic_quotes_gpc` este activată în versiunea limbajului PHP instalată în sistemul dumneavoastră, scriptul nu va funcționa în mod corespunzător.

4. Alocați un timp studiului paginii HTML. Acordați o atenție specială expresiei regulate specificate ca valoare prestabilită a casetei cu text denumite `model`. Această expresie regulată aproximează forma unei adrese de e-mail. Cu toate acestea, nu este o reprezentare perfectă. Căutarea expresiei regulate perfecte pentru reprezentarea adreselor corecte de e-mail este asemănătoare cu expedițiile medievale întreprinse pentru găsirea Sfântului Graal\*. Puteți găsi și alte potențiale expresii regulate la adresa <http://www.phpbuilder.com> și în manualul PHP adnotat.

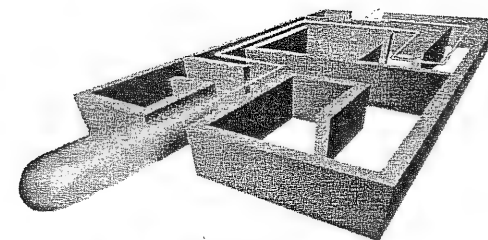
5. Orientați un browser Web spre adresa URL a fișierului HTML încărcat anterior. Ecranul browserului trebuie să fie asemănător celui prezentat în ilustrația următoare. Introduceți o valoare a șirului și, dacă doriți, înlocuiți valoarea expresiei regulate, după care executați clic pe butonul „Trimite interogarea”.
6. La execuția scriptului, acesta încearcă să stabilească o echivalență între expresia regulată și un sub-șir al șirului și afișează rezultatul. Un rezultat caracteristic este prezentat alăturat.



\* În original *Holy Grail* (Sfântul Graal) – În legendele anglo-saxone, un potir folosit de Iisus la Cina cea de Taină și în care Iosif din Arimateea a cules ultimele picături din sângele lui Iisus răstignit pe Cruce; folosit frecvent ca simbol al purității creștine sau ca răsplată a acestuia. – N.T.

### ✓ Test de evaluare

1. Scrieți un șir de formatare care specifică o valoare șir aliniată la stânga, care trebuie să ocupe 24 de spații, urmată de o valoare de tip double aliniată la stânga, cu două cifre zecimale.
2. Scrieți o secvență escape care reprezintă caracterul a cărui valoare ASCII este 45 în octal.
3. Scrieți un apel de funcție și o atribuire care stochează în variabila \$r valoarea variabilei \$s și care elimină caracterele de tip spațiu alb de la început și de la sfârșit.
4. Scrieți un apel de funcție care returnează un șir asemănător cu \$s, dar ale cărui n caractere, numărate de la poziția 1, sunt înlocuite prin șirul \$r.
5. Scrieți o expresie regulată care corespunde numai sub-șirurilor "axb", "ayb" și "azb" care apar la sfârșitul unui șir subiect.



## Partea a III-a: Lucrul cu date stocate

### Modulul 10: Utilizarea variabilelor cookie

#### Scopuri

- Învățați care este modul de funcționare a variabilelor cookie
- Învățați să creați, să obțineți accesul la variabilele cookie și să le ștergeți
- Învățați să stocați mai multe valori într-o variabilă cookie
- Învățați să specificați opțiunile dintr-o variabilă cookie

Acest modul vă prezintă noțiunile introductive referitoare la variabilele de date cookie, o caracteristică HTTP care vă permite să stocați date în sistemul unui utilizator. Variabilele cookie sunt utile pentru stocarea preferințelor utilizatorilor și a altor informații care trebuie reținute atunci când utilizatorul trece la o nouă pagină Web.

#### Accesul la variabilele cookie și crearea acestora

Valorile majorității variabilelor dispar atunci când scriptul PHP care le conține își încheie execuția. Spre deosebire de acestea, valorile variabilelor cookie se pot păstra un timp indefinit. Pentru ca valorile lor să se poată păstra, browserul utilizatorului stochează variabilele cookie în unitatea de hard-disc locală a utilizatorului.

Variabilele cookie sunt utile dintr-o mulțime de puncte de vedere. De exemplu, multe situri Web folosesc variabile cookie pentru a stoca identitatea utilizatorului și

preferințele de vizualizare ale acestuia. Când utilizatorul revine la situl Web, variabilele cookie permit browserului să recunoască utilizatorul și să restaureze opțiunile sitului selectate de către utilizator.

Din păcate, variabilele cookie nu constituie soluția perfectă pentru un mediu de stocare pe termen lung și prezintă o serie de dezavantaje. De exemplu:

- Un utilizator poate dezactiva variabilele cookie prin stabilirea unei opțiuni a browserului
- În anumite situații, variabilele cookie pot fi vizualizate de alți utilizatori decât utilizatorul ale cărui date le stochează
- Un sit poate stoca numai 20 de variabile cookie și numai 4KB de informații în unitatea de hard-disc locală a utilizatorului
- Numeroase versiuni ale browserelor frecvent folosite au erori care le împiedică să folosească variabilele cookie în mod adecvat

În ciuda acestor dezavantaje, variabilele cookie rămân cea mai populară tehnică pentru obținerea unui mediu de stocare pe termen lung. Deci, este important să înțelegi care este modul de funcționare și de utilizare a acestora.

## Accesul la o variabilă cookie

Poate că trăsătura cea mai caracteristică a variabilelor cookie o constituie comoditatea. Dacă ai creat o variabilă cookie, valoarea acesteia este automat pusă la dispoziție ca variabilă PHP având același nume cu acela al variabilei cookie. De exemplu, să presupunem că ai creat o variabilă cookie denumită `fruct` și că îi atribuie valoarea `banana`. Această pereche nume-valoare este apoi pusă la dispoziția fiecărui script PHP asociat sitului dumneavoastră de Web. Deci puteți afișa valoarea variabilei cookie folosind următoarea instrucțiune:

```
echo "Valoarea variabilei cookie este $fruct";
```

Această instrucțiune are ca efect afișarea următorului rezultat:

```
Valoarea variabilei cookie este banana.
```

Variabila PHP de tip tablou asociativ `HTTP_COOKIE_VARS` conține numele și valoarea fiecărei variabile cookie curentă. Dacă doriți să vizualizați fiecare variabilă cookie disponibilă și valoarea acesteia, puteți invoca funcția `phpinfo()`, care afișează valoarea tabloului `HTTP_COOKIE_VARS`. Dacă doriți să obțineți acces la tablou prin metode programatice, puteți folosi un program ca următorul:

```
foreach ($HTTP_COOKIE_VARS as $nume => $valoare)
 echo "
$nume => $valoare";
```

## Crearea unei variabile cookie

Crearea unei variabile cookie este aproape la fel de simplă ca și obținerea accesului la aceasta. Pentru a crea o variabilă cookie, invocați funcția `setcookie()`, care are următoarea formă:

```
setcookie(nume, valoare, expirare)
```

Argumentul `nume` specifică numele variabilei cookie, iar argumentul `valoare` specifică valoarea variabilei. Argumentul `expirare` indică momentul expirării variabilei cookie; după ora specificată, variabila cookie nu mai este accesibilă.

În general, este convenabil să se specifice momentul expirării folosind funcția `time()`, care returnează intervalul de timp (exprimat în secunde) scurs de la 1 ianuarie 1970. Puteți adăuga o valoare de tip decalaj (offset), care specifică intervalul de timp pe durata căruia variabila cookie trebuie să fie accesibilă. De exemplu, să luăm în considerare următoarea instrucțiune:

```
setcookie("fruct", "banana", time()+3600);
```

Această instrucțiune creează o variabilă cookie denumită `fruct`, care are valoarea `banana`. Variabila cookie va fi disponibilă timp de o oră (3600 secunde) de la crearea sa.

Dacă preferați, puteți specifica momentul expirării folosind funcția `mktime()`. Această funcție are următoarea formă:

```
mktime(ore, minute, secunde, luna, zi, an)
```

De exemplu, următoarea instrucțiune creează o variabilă cookie care expiră la o secundă după miezul nopții primei zile a anului 2005:

```
setcookie("fruct", "banana", mktime(0, 0, 1, 1, 1, 2005));
```

## Avertisment

Valorile variabilelor cookie sunt trimise de către browser ca parte a antetelor HTTP. Ca atare, valorile variabilelor cookie trebuie să fie stabilite anterior expedierii oricărui altor valori către browser. Trimiterea fie și a unui singur spațiu vă poate împiedica să configurați valoarea unei variabile cookie. Pentru a evita problemele, asigurați-vă că un script PHP care stabilește o valoare a unei variabile cookie este plasat în partea superioară a fișierului, fără caractere de tip spațiu alb care să-l preceadă. De asemenea, stabiliți valoarea variabilei cookie înainte de a executa o instrucțiune `echo` sau o altă instrucțiune PHP care trimite browserului date de ieșire.



## Sfatul specialistului

**Întrebare:** Ce este un antet HTTP?

**Răspuns:** Înainte de a trimite date HTML unui browser, un server Web trimite, în general, unul sau mai multe antete HTTP; aceste antete sunt cunoscute

sub numele de antete de răspuns al serverului. Similar, înainte de a trimite informații unui server Web, un browser Web trimite, în general, unul sau mai multe antete HTTP; aceste antete sunt cunoscute sub numele de antete de cerere.

Antetele de răspuns al serverului frecvent folosite descriu configurația serverului și furnizează informații referitoare la adresa URL solicitată de client. Antetele de cerere utilizate de obicei descriu configurația clientului și formatele de date acceptabile de către client.

În afară de antetele de răspuns al serverului și de antetele de cerere, protocolul HTTP folosește antete generale și antete de entitate. Antetele generale sunt folosite atât de către clienți, cât și de către servere, pentru a specifica informații precum data curentă și opțiunile de conexiune. Antetele de entitate descriu formatul datelor schimbate de un client și un server.

## Ștergerea unei variabile cookie

Deoarece o variabilă cookie are o dată de expirare, aceasta va fi ștersă automat la un oarecare interval de timp după crearea sa. Totuși, puteți șterge o variabilă cookie imediat. Pentru aceasta, fixați momentul expirării variabilei cookie la un moment de timp din trecut. De exemplu, pentru a șterge o variabilă cookie denumită `fruct`, puteți folosi următoarea instrucțiune:

```
setcookie("fruct", "", time()-3600);
```

Această instrucțiune stabilește timpul de expirare cu o oră (3600 de secunde) în urmă. Remarcați că valoarea variabilei cookie este exprimată sub forma unui șir vid; din moment ce variabila cookie nu va mai fi disponibilă, valoarea sa nu mai are importanță.



### Test „la minut”

- Care este variabila PHP folosită pentru a include numele fiecărei variabile cookie disponibile?
- Care este funcția utilizată pentru crearea unei variabile cookie?
- Care este funcția folosită pentru ștergerea unei variabile cookie?

Răspunsuri la test:

- `$HTTP_COOKIE_VARS` `setcookie()` | `setcookie()`
- `setcookie()`
- `setcookie()`

## Tehnici avansate de utilizare a variabilelor cookie

Această secțiune prezintă unele tehnici mai avansate pentru lucrul cu variabile cookie. Prima sub-secțiune explică modul de stocare a mai mult de 20 de valori într-o singură variabilă cookie. Cea de-a doua sub-secțiune explică modul de utilizare a mai multor argumente suplimentare ale funcției `setcookie()`.

### Stocarea mai multor valori într-o variabilă cookie

Deoarece un sit Web poate stoca numai 20 de variabile cookie în sistemul unui utilizator, capacitatea de a stoca mai multe valori într-o singură variabilă cookie este utilă. În conformitate cu manualul PHP pe suport electronic, puteți realiza acest lucru prin specificarea unui tablou ca nume al variabilei cookie. De exemplu, puteți folosi un program ca acesta:

```
// Un exemplu eronat de variabila cookie cu mai multe valori
for ($i = 0; $i < 30; $i++)
{
 setcookie("cookies[$i]", "$i");
}

(isset($cookies))
{
 foreach ($cookies as $i => $cookie)
 {
 echo "
$i => $cookie";
 }
}
```

Din păcate, acest procedeu *nu* funcționează. Contrar informațiilor din manualul PHP, fiecare element al tabloului este stocat într-o variabilă cookie separată. Astfel, prin utilizarea acestui procedeu nu puteți stoca mai mult de 20 de valori.

Pe de altă parte, stocarea mai multor valori într-o singură variabilă cookie *este* posibilă. Pentru aceasta, inserați valorile într-un tablou și folosiți funcția `serialize()` pentru a „împacheta” elementele tabloului într-un șir; ulterior, puteți recupera valoarea tabloului folosind funcția `unserialize()`. Iată un exemplu care prezintă modul de creare a unei variabile cookie care conține mai multe valori, precum și modul de acces la aceasta:

```
// Se creeaza un tablou
for ($i = 0; $i < 30; $i++)
{
 $tablou[$i]=$i;
}

// Se impacheteaza intregul tablou intr-un sir
$s = serialize($tablou);
```

```
// Se creeaza o variabila cookie si se stabileste valoarea sa
setcookie("cookies", $s);
```

```
if(isset($cookies))
{
 // Se despacheteaza valoarea variabilei cookie
 $tablou = unserialize(stripslashes($cookies));

 // Demonstreaza ca totul este in ordine,
 // prin afisarea elementelor tabloului
 foreach($tablou as $i => $cookie)
 {
 echo "
$i => $cookie";
 }
}
```

Funcția `stripslashes()` este folosită pentru eliminarea secvențelor escape adăugate la șir atunci când valoarea variabilei cookie este returnată de PHP.

### Avertisment

Deși acest procedeu reușește să ocolească limita de celor 20 de variabile cookie, nu poate depăși limita celor 4KB de date stocate într-o variabilă cookie pentru fiecare sit Web în parte. Dacă doriți să stocați mai mult de 4KB de date, trebuie să stocați datele într-o bază de date pe parte de server sau într-un alt loc decât o variabilă cookie.

## Specificarea accesului la o variabilă cookie și alte opțiuni

Funcția `setcookie()` poate prelua maximum șase argumente, inclusiv trei argumente despre care nu am discutat încă. Iată formatul complet al funcției `setcookie()`:

```
setcookie(ume, valoare, expirare, cale, domeniu, sigur)
```

Argumentele `ume`, `valoare` și `expirare` au fost descrise în secțiunea precedentă a acestui modul.

Argumentul `cale` vă permite să specificați calea URL asociată variabilei cookie. În mod prestabilit, variabila cookie este disponibilă pentru scripturile din catalogul care conține scriptul în care a fost configurată variabila respectivă, precum și pentru scripturile din sub-catalogele aferente catalogului respectiv. În particular, scripturilor din catalogele părinte ale catalogului care conține scriptul nu li se permite accesul prestabilit la variabila cookie.

Pentru a pune variabila cookie la dispoziția scripturilor dintr-un anumit catalog și din sub-catalogele sale, specificați o valoare a argumentului `cale`. De exemplu, pentru a pune variabila cookie la dispoziția întregului arbore de cataloge, specificați `"/` ca valoare a argumentului `cale`; pentru a face variabila cookie disponibilă în catalogul `/-test` și în sub-catalogele sale, specificați `"/-test/"` ca valoare a argumentului `cale`.

### Avertisment

O complicație în utilizarea argumentului `cale` o constituie modalitatea de identificare a numelor catalogelor. Specificând `"/-test/"` ca valoare a argumentului `cale`, variabila cookie va deveni disponibilă în `/-test1`, `/-test2` și în toate catalogele cu nume similare, pe lângă catalogul `/-test` și sub-catalogele sale.

Dacă nu este specificat nici un argument `domeniu`, o variabilă cookie este disponibilă numai pentru scripturile rezidente pe serverul Web care a creat variabila respectivă. Argumentul `domeniu` vă permite să specificați numele de domeniu asociat unei variabile cookie. În consecință, variabila cookie va fi disponibilă numai pentru siturile Web din cadrul domeniului specificat. De exemplu, să presupunem că un script din serverul Web `http://www.subdomeniu.domeniu.com` creează o variabilă cookie. În mod prestabilit, variabila cookie este disponibilă numai pentru gazda respectivă. Cu toate acestea, puteți face variabila cookie disponibilă pe întreg domeniul `subdomeniu.domeniu.com`, specificând `"subdomeniu.domeniu.com"` ca valoare a argumentului `domeniu`.

### Sugestie

Specificația Netscape pentru variabile cookie ([http://www.netscape.com/newsref/std/cookie\\_spec.html](http://www.netscape.com/newsref/std/cookie_spec.html)) impune ca argumentul `domeniu` să conțină minimum două caractere punct. Ca atare, nu trebuie să specificați un șir de tipul `"domeniu.com"` ca valoare a argumentului `domeniu`.

Argumentul `sigur` este o valoare întreagă, care specifică dacă variabila cookie trebuie trimisă prin intermediul unei conexiuni sigure (HTTPS). Specificați valoarea 1 pentru a împiedica transmiterea variabilei cookie în cazul în care conexiunea nu este sigură; pentru a permite transmiterea variabilei cookie prin conexiuni HTTP obișnuite, specificați valoarea 0.



### Sfatul specialistului

**Întrebare:** Dacă o persoană cu acces la unitatea de hard-disc a utilizatorului poate citi valorile stocate într-o variabilă cookie, cum se poate păstra confidențialitatea informațiilor stocate în această variabilă?

**Răspuns:** Deoarece browserele stochează variabilele cookie în unitatea de hard-disc locală, utilizatorii unui sistem pot obține accesul la fișierele cookie și pot citi sau chiar modifica informațiile conținute în fișierele respective. O modalitate de a preveni situația prezentată constă în criptarea datelor stocate în variabilele cookie. Pentru aceasta, puteți folosi funcțiile `Mcrypt` din PHP. Funcțiile în cauză sunt incluse în biblioteca `libmcrypt`, care nu face parte din versiunea insta-



lată în mod prestabilit a limbajului PHP; deci, este posibil să fiți obligat a solicita administratorului dumneavoastră de sistem să instaleze biblioteca și să configureze PHP astfel încât să fie capabil să obțină accesul la aceasta. Pentru informații despre instalarea și utilizarea funcțiilor Mcrypt, vezi adresa <http://www.php.net>.

### Avertisment

Deși argumentele expirare și cale ale funcției `setcookie()` sunt opționale, unele versiuni ale principalelor browsere prezintă erori care le determină să refuze variabilele cookie dacă aceste argumente nu sunt specificate. Ca atare, în general este recomandat să specificați aceste argumente.



### Test „la minut”

- Care este numărul de valori care pot fi asociate unui fișier cookie?
- Care este funcția folosită pentru codificarea unui tablou astfel încât unei variabile cookie să-i poată fi asociate mai multe valori?
- Pentru a indica faptul că o variabilă cookie trebuie să fie trimisă numai prin intermediul unei conexiuni sigure (HTTPS), care este valoarea pe care trebuie să o primească argumentul sigur al funcției `setcookie()`?



### Proiect 10-1: O pagină de deschidere a sesiunii de lucru

În cadrul acestui proiect, veți construi o pagină HTML și un script PHP care permit unui utilizator să deschidă sesiunea de lucru cu un sistem. Pagina permite utilizatorului să introducă un identificator de utilizator și o parolă, care sunt autentificate de script prin confruntarea cu un set stocat de identificatori de utilizator și parole corecte. Dacă autentificarea utilizatorului reușește, scriptul configurează o variabilă cookie care indică scripturilor ulterioare că utilizatorul a fost autentificat.

### Scopurile proiectului

- Prezentarea unei aplicații a variabilelor cookie
- Prezentarea modului de creare a unei pagini simple de deschidere a sesiunii de lucru și a unui script

### Pas cu pas

1. Plasați următorul script PHP într-un fișier numit `p-10-1.php` și încărcați acest fișier în serverul dumneavoastră PHP:

### Răspunsuri la test:

- 20
- `serialize()`
- 1

```
<?
$parole = array("mihai" =>"portocala",
 "stefan" =>"cartof",
 "andrei" =>"arahida");
if ($parola == $parole[$nume_utilizator])
{
 setcookie("nume_utilizator", $nume_utilizator, time()+1200);
 echo "<H2>Accesul este permis.</H2>";
}
else
{
 setcookie("nume_utilizator", "", time()-3600);
 echo "<H2>Nume de utilizator sau parola incorecte: accesul interzis.</H2>";
}
?>
```

2. Plasați următorul text HTML într-un fișier denumit `p-10-1.html` și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul `p-10-1.php`:

```
<HTML>
<HEAD>
<TITLE>Proiect 10-1</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="p-10-1.php">
<H2>Proiect 10-1: Pagina de login</H2>

Numele utilizatorului:

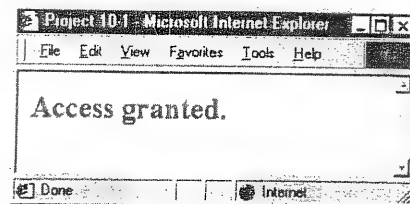
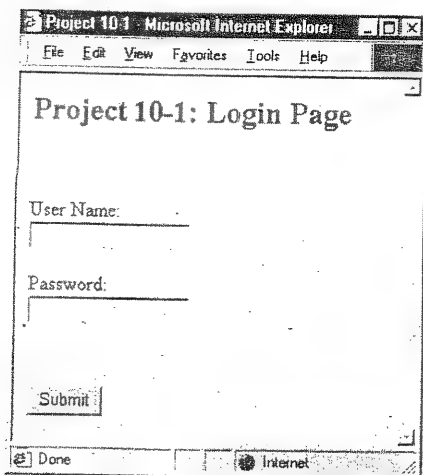
<INPUT TYPE="TEXT" NAME="nume_utilizator" SIZE="16">

Parola:

<INPUT TYPE="PASSWORD" NAME="parola" SIZE="16">

<INPUT TYPE="SUBMIT" VALUE="Trimite">
</FORM>
</BODY>
</HTML>
```

3. Alocați un timp studiului scriptului PHP. Remarcați că acesta autentifică datele introduse de utilizator comparându-le cu valorile stocate într-un tablou. Într-o aplicație din lumea reală, valorile corecte sunt, mai probabil, stocate într-o bază de date. Într-un modul ulterior veți învăța care este modul de lucru cu bazele de date.
4. De asemenea, remarcați că variabila cookie este configurată astfel încât să expire în 20 de minute. Scripturile ulterioare pot reinițializa momentul expirării variabilei cookie, astfel încât autentificarea utilizatorului să nu fie anulată în timp ce acesta lucrează. Totuși, dacă utilizatorul încetează să mai folosească sistemul, momentul expirării variabilei cookie va garanta că utilizatorul este automat scos din sistem după 20 de minute. Aceasta contribuie la evitarea breșelor de securitate generate de utilizatori care părăsesc o sesiune de lucru.
5. Orientați un browser Web spre adresa URL a fișierului HTML încărcat anterior. Ecranul browserului trebuie să fie asemănător celui prezentat pe pagina următoare, în stânga. Introduceți un identificator de utilizator și o parolă și executați clic pe butonul „Trimite”.
6. Scriptul verifică datele pe care le-ați introdus. Dacă ați introdus un identificator de utilizator și o parolă corecte, trebuie să vedeți un ecran asemănător celui prezentat pe pagina următoare, în dreapta; în caz contrar, veți vedea un ecran care conține un mesaj de eroare.



### Test de evaluare

1. Scrieți o instrucțiune PHP care creează o variabilă cookie denumită corect, care are valoarea "false"; stabiliți ca variabila cookie să expire în 30 de minute.
2. Scrieți o instrucțiune PHP care șterge o variabilă cookie denumită trecut.
3. Scrieți o instrucțiune PHP care afișează valoarea variabilei cookie denumite varsta.
4. Scrieți o instrucțiune PHP care împachetează tabloul numit \$continut într-un șir denumit \$x.
5. Scrieți o instrucțiune PHP care creează o variabilă cookie numită oriunde, care are valoarea "aici". Variabila cookie trebuie să expire în 30 de minute și trebuie să fie accesibilă în fiecare catalog al arborelui Web.

## Modulul 11: Lucrul cu fișiere și cataloage

### Scopuri

- Învățați modul de funcționare a sistemului de fișiere UNIX
- Învățați să obțineți informații despre fișiere și cataloage
- Învățați să citiți și să scrieți fișiere
- Învățați să configurați permisiuni de fișiere și cataloage
- Învățați să încărcați, să copiați, să ștergeți și să modificați denumirea fișierelor
- Învățați să creați și să ștergeți cataloage
- Învățați să citiți cataloage și să navigați în acestea

Acest modul explică facilitățile oferite de PHP pentru lucrul cu fișiere și cataloage. Fișierele și cataloagele vă permit să stocați date în server, astfel încât datele să poată fi reținute și să fie accesibile de către mai mulți utilizatori.

### Sistemul de fișiere UNIX

Pentru a înțelege cum trebuie utilizat limbajul PHP pentru a lucra cu fișiere și cataloage, trebuie să înțelegeți sistemul de fișiere UNIX. Acest fapt este valabil chiar dacă folosiți PHP sub Microsoft Windows, deoarece modelul folosit de PHP pentru lucrul cu fișiere și cataloage este bazat pe UNIX.

Această secțiune explică sistemul de fișiere UNIX și modul de utilizare a comenzilor UNIX pentru lucrul cu fișiere și cataloage. Dacă lucrați cu Microsoft Windows, în general comenzile date în această secțiune nu vor funcționa corect. Totuși, în mediul respectiv veți lucra cu fișiere și cataloage folosind cu precădere Windows Explorer, nu DOS. Deci probabil că nu este necesar să cunoașteți comenzile DOS similare comenzilor UNIX explicate în această secțiune.

### Sugestie

Pentru a putea lucra cu fișiere și cataloage, trebuie să deschideți sesiunea de lucru pe sistemul unde sunt rezidente acestea. Dacă nu aveți acces la o consolă locală, puteți avea acces la sistem prin intermediul Telnet sau SSH, două protocoale Internet cu o largă utilizare. Pentru a afla care este modul de acces la fișierele și cataloagele unui sistem, apăsați la administratorul sistemului respectiv.

## Lucrul cu fișiere UNIX

Un *fișier* este o serie de octeți stocați pe o unitate de hard-disc, CD-ROM sau alt mediu. Fișierele primesc nume pentru a se putea face cu ușurință referire la acestea. Un nume de fișier UNIX poate avea o lungime aproape nelimitată și poate include aproape orice caracter. Totuși, se recomandă utilizarea unor nume de fișiere care să fie suficient de scurte pentru a putea fi tastate cu ușurință și care să includă numai caractere vizibile, care nu au nici o semnificație specială pentru interpretorul UNIX. Spre deosebire de numele de fișiere Microsoft Windows, numele de fișiere UNIX sunt sensibile la diferența între majuscule și minuscule; ca atare, a și A se referă la fișiere UNIX diferite. Pentru a evita problemele, mai ales atunci când deplasați fișiere între UNIX și Windows, trebuie să folosiți minuscule, cifre, puncte, caractere de subliniere și cratime în numele fișierelor; de asemenea, numele fișierelor trebuie să înceapă cu o minusculă sau cu o cifră.

### Notă

Un octet este aproximativ identic cu un caracter. Totuși, limbile (nu limbajele) care conțin în alfabetul lor numeroase caractere pot necesita mai mulți octeți pentru reprezentarea unui caracter. Deseori, această diferență este lipsită de importanță.

## Vizualizarea informațiilor despre fișiere

Pentru a vizualiza informații care descriu un fișier, emiteți comanda

```
ls -l nume_fisier
```

unde *nume\_fisier* este numele fișierului. Figura 11-1 prezintă datele de ieșire caracteristice ale comenzii *ls*.

Datele de ieșire includ următoarele câmpuri:

- Tipul fișierului și permisiuni: tipul fișierului și permisiunile de acces. Aceste caracteristici ale fișierului sunt descrise în sub-sectiunile intitulate „Tipuri de fișiere” și „Privilegii de fișier”.
- Legături: numărul legăturilor hard asociate acestui fișier. Fiecare legătură hard stabilește un nume după care este cunoscut fișierul respectiv. În general, puteți ignora acest câmp.
- Utilizator: Numele utilizatorului care este posesorul fișierului.
- Grup: Numele grupului care este posesorul fișierului.
- Dimensiunea fișierului: dimensiunea fișierului, exprimată în octeți.
- Data modificării: data și ora ultimei modificări a fișierului. Dacă fișierul nu a suferit modificări recente, vor fi afișate data și anul.
- Numele fișierului: numele atribuit fișierului.

## Vizualizarea unui fișier

Pentru a vizualiza conținutul unui fișier, emiteți comanda

```
more nume_fisier
```

unde *nume\_fisier* este numele fișierului. În cazul în care fișierul conține mai multe linii decât poate accepta ecranul sau fereastra, comanda *more* afișează numai numărul liniilor care se încadrează în ecran, respectiv fereastră. Pentru a parcurge fișierul înainte, apăsați pe tasta spațiu. Pentru a parcurge fișierul în sens invers, tastați litera *b*. Pentru a părăsi comanda, tastați litera *q*.

-rw-r--r--	1	root	root	86	Mar 25 15:08	networks.txt
Tip și permisiuni	Legături	Utilizator	Grup	Dimensiunea fișierului	Data modificării	Numele fișierului

Figura 11-1 Datele de ieșire ale comenzii *ls*

### Sugestie

Datele de ieșire ale comenzii *more* vor fi inteligibile numai dacă fișierul conține date în format ASCII. Pentru a vizualiza un fișier binar, puteți folosi comanda *od*.

## Editarea unui fișier

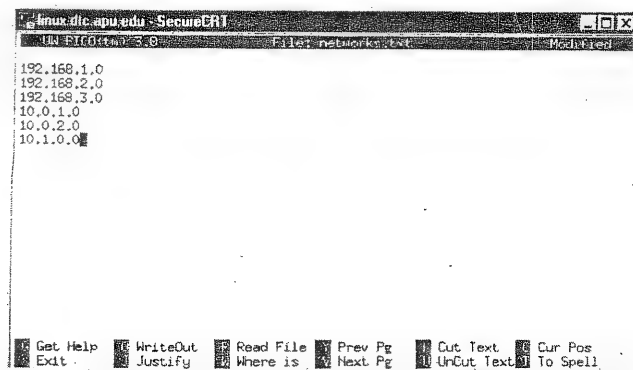
UNIX acceptă o varietate de editoare pe care le puteți folosi pentru a edita un fișier. Întrebați-l pe administratorul de sistem care sunt editoarele disponibile în sistemul dumneavoastră. Un editor preferat de numeroși începători este Pico, editor asociat cu popularul program client de e-mail Pine. Pico include documente de asistență încorporate și este ușor de învățat și utilizat. De asemenea, datorită asocierii sale cu Pine, Pico se găsește pe numeroase sisteme UNIX.

Pentru a edita un fișier folosind Pico, emiteți comanda

```
pico nume_fisier
```

unde *nume\_fisier* este numele fișierului pe care doriți să-l editați. Dacă doriți să creați un fișier nou, pur și simplu omiteți numele fișierului. Ecranul editorului Pico se prezintă ca pe pagina următoare.

Pico prezintă numeroase comenzi utile, afișându-le în ultimele două linii ale ecranului său. Fiecare comandă este emisă menținând apăsată tasta CTRL și apăsând o tastă literală. Pico notează această convenție prin prefixarea literei care simbolizează comanda cu un caracter de tip accent circumflex (^). Tabelul 11-1 descrie numeroase comenzi Pico utile.



### Ștergerea unui fișier

Pentru a șterge un fișier, emiteți comanda

```
rm nume_fisier
```

unde *nume\_fisier* este numele fișierului.

### Atenție

Spre deosebire de Windows, UNIX nu salvează, în general, fișierele șterse într-un așa-zis „recipient de reciclare” (Recycle Bin). Ca atare, ștergerea unui fișier UNIX este, în general, un act irevocabil.

### Copierea unui fișier

Pentru a copia un fișier, emiteți comanda

```
cp fisier_vechi fisier_nou
```

**Tabelul 11-1** Comenzi utile ale editorului Pico

Comandă	Descriere
CTRL-C	Afișează poziția curentă a cursorului (numărul liniei și al coloanei)
CTRL-G	Afișează documentele de asistență Pico.
CTRL-J	Aliniază paragraful curent.
CTRL-K	Taie linia curentă.
CTRL-O	Scrie pe disc conținutul bufferului de editare.
CTRL-R	Citește un fișier în bufferul de editare.
CTRL-T	Lansează utilitarul de verificare ortografică al editorului Pico.
CTRL-U	Lipește text.
CTRL-V	Deplasează textul înainte.
CTRL-W	Caută text.
CTRL-X	Părăsește programul Pico. Programul va afișa un prompt dacă bufferul de editare nu a fost salvat.
CTRL-Y	Deplasează textul înapoi.

unde *fișier\_vechi* este numele fișierului pe care doriți să-l copiați (fișierul sursă), iar *fișier\_nou* este numele pe care doriți să-l repartizați copiei (fișierul destinație). Comanda *cp* nu afectează fișierul sursă.

### Atenție

În funcție de configurația unui sistem UNIX, este posibil ca prin comanda *cp* să se suprascrie un fișier existent. Nu uitați să evitați suprascrierea accidentală a unui fișier important.

### Modificarea numelui unui fișier

Pentru a modifica numele unui fișier, emiteți comanda

```
mv fisier_vechi fisier_nou
```

unde *fișier\_vechi* este numele curent al fișierului, iar *fișier\_nou* este numele dorit.

### Sugestie

Unele sisteme UNIX interzic utilizatorilor să modifice numele fișierului de la un dispozitiv sau partiție la alta. Pentru a afla care sunt restricțiile în vigoare în sistemul dumneavoastră, luați legătura cu administratorul de sistem.

### Tipurile fișierelor

Figura 11-1 a prezentat datele de ieșire caracteristice ale comenzii *ls*. Primul câmp al datelor de ieșire indică tipul fișierului și privilegiile asociate acestuia. Primul caracter al câmpului indică tipului fișierului. Între valorile posibile se numără următoarele:

Tip	Descriere
-	Fișier normal
b	Fișier de dispozitiv, incapabil de transferuri în bloc
d	Catalog
l	Legătură simbolică
p	Canal denumit (fifo)
s	Soclu

Cele mai importante tipuri de fișiere sunt fișierul normal și catalogul. Celelalte tipuri de fișiere au destinații care nu necesită atenția noastră imediată.

### Proprietatea asupra fișierelor

Fiecare fișier are un cont de utilizator asociat, cunoscut sub numele de *posesor* al fișierului. Puteți determina posesorul unui fișier prin emiterea comenzii *ls*. Utilizatorul care creează un fișier devine posesorul fișierului. Cu toate acestea, un administrator de sistem poate atribui un fișier unui alt utilizator, prin emiterea comenzii *chown*.

Administratorii de sistem UNIX pot defini *grupuri de utilizatori*, sau pur și simplu *grupuri*, care reprezintă seturi de utilizatori. Un utilizator poate fi membru al unui număr oricât de mare de grupuri.

Fiecare fișier are un grup asociat, cunoscut sub numele de *grupul posesor* al fișierului. Puteți determina grupul posesor al unui fișier prin emiterea comenzii `ls`. Unele sisteme UNIX configurează în mod automat un grup privat asociat fiecărui utilizator. Pe asemenea sisteme, posesorul și grupul posesor al unui fișier au, în mod normal, același nume.

Posesorul unui fișier poate atribui unui fișier un nou grup posesor prin emiterea comenzii `chgrp`, care are următoarea formă:

```
chgrp grup nume_fisier
```

unde *nume\_fisier* este numele fișierului, iar *grup* este numele grupului. În afară de calitatea de posesor al fișierului, utilizatorul care emite comanda trebuie să fie un membru al grupului *grup*.

## Privilegii de fișier

Privilegiile asociate unui fișier determină operațiile pe care utilizatorii le pot efectua cu fișierul respectiv. Puteți determina privilegiile asociate unui fișier prin emiterea comenzii `ls`. Așa cum se poate vedea în figura 11-1, primul câmp din datele de ieșire ale comenzii `ls` indică tipul și privilegiile unui fișier. Primul caracter al câmpului indică tipul fișierului; celelalte două indică privilegiile.

Privilegiile sunt date sub forma a trei grupuri alcătuite din câte trei caractere fiecare; cu alte cuvinte, trei triade. Prima triadă indică privilegiile acordate posesorului fișierului. Cea de-a doua triadă indică privilegiile acordate membrilor grupului care este posesorul fișierului. Cea de-a treia triadă indică privilegiile acordate altor utilizatori, cu alte cuvinte, persoanelor care nu sunt nici posesoare ale fișierului și nici nu sunt membre ale grupului care este posesorul fișierului. De exemplu, să presupunem că primul câmp al datelor de ieșire ale comenzilor `ls` se prezintă astfel:

```
-rwxr-xr--
```

Ignorând primul caracter, care reprezintă tipul fișierului, aceste date de ieșire reflectă următoarele privilegii:

- posesor, `rwx`
- membru al grupului, `r-x`
- alte persoane, `r--`

Fiecare caracter al unei triade de privilegii poate fi o literă sau o cratimă. Literele au următoarele semnificații:

- `r`, fișierul poate fi citit
- `w`, se poate scrie în fișier
- `x`, conținutul fișierului poate fi executat

## Notă

Privilegiul `x` este semnificativ numai pentru fișierele care includ un conținut executabil, cum sunt fișierele binare executabile sau anumite categorii de scripturi.

Caracterele unei triade apar întotdeauna în secvența `rwx`. Dacă o anumită literă este înlocuită de o cratimă, privilegiul asociat nu este utilizabil. De exemplu, să examinăm privilegiile specificate anterior:

```
-rwxr-xr--
```

Aceste caractere au următoarea semnificație:

- `rwx`, posesorul fișierului poate citi, scrie sau executa fișierul
- `r-x`, membrii grupului posesor al fișierului pot citi sau executa fișierul, dar nu pot scrie în fișier
- `r--`, alți utilizatori pot citi fișierul, dar nu pot scrie în fișier sau executa conținutul fișierului

Posesorul unui fișier poate modifica privilegiile asociate fișierului emițând comanda `chmod`. Această comandă are două forme. O formă vă permite să specificați privilegiile folosind cifre scrise în octal; cealaltă vă permite să le specificați folosind litere.

Pentru a specifica privilegiile folosind cifre în octal, calculați valoarea numerică a fiecărei triade. Pentru aceasta, însumați numerele corespunzătoare fiecărui privilegiu disponibil din cadrul triadei. Numerele asociate privilegiilor sunt următoarele:

Privilegiu	Valoare
<code>r</code>	4
<code>w</code>	2
<code>x</code>	1

De exemplu, privilegiul `rwx` are valoarea  $4 + 2 + 1 = 7$ . Similar, privilegiul `r-x` are valoarea  $4 + 1 = 5$ , iar privilegiul `r--` are valoarea 4. După ce ați calculat valoarea numerică a fiecărei triade, formați un număr din trei cifre scris în octal, care este alcătuit din valoarea numerică a privilegiilor utilizatorilor, valoarea numerică a privilegiilor membrilor grupului, respectiv valoarea numerică a privilegiilor altor utilizatori. De exemplu, privilegiile `rwxr-xr--` corespund valorii în octal 754.

Forma comenzii `chmod` care folosește cifre în octal este următoarea:

```
chmod mod nume_fisier
```

unde *mod* este numărul din trei cifre scris în octal care indică privilegiile, iar *nume\_fisier* este numele fișierului căruia urmează a i se aplica privilegiile. De exemplu, pentru a acorda posesorului acces complet la fișierul `test` și pentru a acorda altor utilizatori numai acces de citire, emiteți comanda:

```
chmod 744 test
```

Majoritatea utilizatorilor găsesc mai comodă utilizarea acelei forme a comenzii `chmod` care le permite specificarea privilegiilor folosind litere. Această formă alternativă vă permite să specificați privilegiu, precum și să adăugați sau să extrageți privilegiu dintr-un fișier. Iată sintaxa formei alternative:

```
chmod utilizatori operatie privilegii
```

Între argumentele comenzii nu sunt permise spații. Argumentul utilizatori include între una și trei dintre următoarele litere:

- `u`, care indică utilizatorul posesor al fișierului
- `g`, care indică pe membrii grupului posesor al fișierului
- `o`, care indică utilizatori alții decât posesorul și membrii grupului posesor

Argumentul `operatie` este unul din următoarele:

- `=`, care arată că privilegiile specificate trebuie să înlocuiască privilegiile existente
- `+`, care arată că privilegiile specificate trebuie extinse
- `-`, care arată că privilegiile specificate trebuie retrase

Argumentul `privilegii` include între una și trei din următoarele litere:

- `r`, fișierul poate fi citit
- `w`, în fișier este permisă scrierea
- `x`, conținutul fișierului poate fi executat

Se pot specifica mai mulți utilizatori, mai multe operații și mai multe grupuri; fiecare specificație trebuie separată de următoarea specificație cu ajutorul unei virgule.

De exemplu, iată o comandă care instituie privilegiile `rxr--r--` pentru fișierul `test`:

```
chmod u=rwx,g=r,o=r test
```

Iată o comandă care retrage privilegiile de scriere fiecărui utilizator, cu excepția posesorului fișierului, respectiv adaugă privilegiile de executare la privilegiile posesorului fișierului:

```
chmod u+x,g-o,w test
```

Această comandă nu modifică nici privilegiile de citire și de scriere ale posesorului fișierului, nici privilegiile de citire și de execuție ale celorlalți utilizatori.

## Notă

Sistemele UNIX furnizează un cont special de utilizator, denumit *rădăcină* (root) sau *super-utilizator* (superuser), care poate obține acces la fișiere și le poate manipula fără nici un fel de restricții. Administratorul unui sistem este, în general, singura persoană autorizată să utilizeze contul rădăcină.



## Sfatul specialistului

**Întrebare:** Când studiez datele de ieșire ale comenzii `ls`, uneori mai apar și alte privilegii în afară de `r`, `w` și `x`. Care este semnificația acestora?

**Răspuns:** Uneori, administratorii de sistem folosesc numeroase privilegii speciale. De exemplu, un privilegiu special, cunoscut sub numele de `setuid`, modifică temporar identitatea utilizatorului deținător al acestui privilegiu care rulează un fișier program. Asemenea privilegiu special nu sunt, în general, folosite de programatorii aplicațiilor PHP.

## Utilizarea cataloagelor UNIX

Pentru a facilita lucrul cu fișiere, UNIX vă permite să le organizați în *cataloage*. În Microsoft Windows, cataloagele sunt cunoscute sub numele de *dosare* (folders). Un catalog poate conține alte cataloage, cunoscute sub numele de *sub-cataloage* acestuia; catalogul are denumirea de *catalog părinte* al fiecăruia dintre sub-cataloagele sale. Cataloagele și sub-cataloagele unui sistem UNIX formează un singur arbore sau ierarhie. Catalogul amplasat cel mai sus în arbore este cunoscut sub numele de *catalog rădăcină* și se scrie folosind simbolul `/`. Toate celelalte cataloage sunt sub-cataloage ale catalogului rădăcină.

În mod caracteristic, catalogul rădăcină are sub-cataloage precum `bin`, `sbin`, `home` și `tmp`. *Calea absolută* a unui catalog este lista cataloagelor (începând de la catalogul rădăcină) care trebuie parcursă pentru a se ajunge la catalog. Fiecare catalog din listă este separat de catalogul următor de un caracter slash orientat înainte (`/`). De exemplu, calea absolută a sub-catalogului `bin` al catalogului rădăcină este `/bin`. În cazul în care catalogul `/bin` ar fi avut un sub-catalog denumit `there`, calea absolută de acces ar fi `/bin/there`.

## Notă

Sistemele Microsoft Windows folosesc caractere slash orientate înapoi (`\`) pentru separarea componentelor unei căi. Cu toate acestea, versiunile pentru Windows ale limbajului PHP sunt capabile de a interpreta corect o cale specificată folosind caractere slash orientate înainte. Cu toate acestea, dacă scrieți un program PHP care prelucrează căi de acces în format Windows, rețineți că un caracter slash orientat înapoi care apare într-un șir PHP poate fi interpretat ca fiind caracterul inițial al unei secvențe escape. Poate fi necesar să înlocuiți fiecare caracter slash orientat înapoi cu o pereche de caractere slash orientate înapoi, pentru a împiedica limbajul PHP să interpreteze șirul în mod eronat.

Joc de cuvinte. În limba engleză, numele catalogului `bin` și forma de participiu trecut a verbului `to be` (a fi), în specia `been`, se pronunță aproximativ asemănător. De aceea, expresia `bin there` se pronunță la fel cu `been there`, adică *am fost acolo*, în engleza americană. — N.T.



Și fișierele pot avea căi absolute. Un fișier numit *donethat*, rezident în catalogul */bin/there*, va avea calea de acces absolută */bin/there/donethat*.\*

În general, un utilizator UNIX are un catalog asociat, cunoscut sub numele de *catalog de bază* al utilizatorului; în mod caracteristic, un catalog de bază este un sub-catalog al catalogului */home*. La orice moment de timp, un program sau un interpretor de comenzi are un catalog asociat, denumit *catalog curent de lucru*. Când un utilizator deschide sesiunea de lucru cu un sistem UNIX, catalogul de bază al utilizatorului este stabilit, în general, drept catalog curent de lucru al sesiunii.

Fișierele și sub-catalogele pot fi desemnate relativ la catalogul curent de lucru, nu numai prin intermediul unei căi absolute. Această formă de referire se numește *cale relativă*. O cale relativă nu începe niciodată cu un caracter slash, deoarece prin slash se înțelege catalogul rădăcină.

De exemplu, să presupunem că */home/bill* este catalogul curent de lucru. Fișierul *test* din catalogul respectiv poate fi desemnat prin intermediul căii absolute */home/bill/test* sau prin calea relativă *test*. Evident, calea relativă este mult mai ușor de tastat și, ca atare, este deseori preferată.

Din nou, să presupunem că */home/bill* este catalogul curent de lucru. În continuare, să presupunem că acest catalog conține sub-catalogul *arhiva*, care include fișierul *note-platite*\*\*. Fișierul poate fi desemnat cu ajutorul căii absolute */home/bill/arhiva/note-platite* sau prin intermediul căii relative *arhiva/note-platite*.

Fiecare catalog are două sub-cataloge speciale, denumite *.* și *..*. Sub-catalogul denumit *.* este un alias al catalogului însuși; sub-catalogul denumit *..* este un alias al catalogului părinte. Puteți folosi aceste sub-cataloge speciale pentru a forma căi relative. De exemplu, să presupunem că */home/bill* este catalogul curent de lucru. Puteți face referire la fișierul */home/test* sub forma *../test*, deoarece simbolul *..* se referă la */home*, catalogul părinte al catalogului */home/bill*.

### Notă

Chiar dacă un catalog rădăcină nu are nici un catalog părinte, conține totuși un catalog cu numele *..*. În această situație specială, catalogul *..* este un alias al catalogului rădăcină.

### Determinarea și modificarea catalogului curent de lucru

Pentru a determina catalogul curent de lucru, emiteți comanda:

```
pwd
```

\* Jocul de cuvinte continuă... S-a format o expresie frecvent folosită în engleza americană, și anume *been there, done that*, cu sensul *am fost acolo, am făcut aia* – N.T.

\*\* Un alt joc de cuvinte. În limba engleză, forma contrasă a prenumelui autorului – în speță *Bill* – și traducerea termenului *notă de plată* – adică *bill* – se scriu și se pronunță absolut la fel. – N.T.

Pentru a înlocui catalogul curent de lucru, emiteți comanda:

```
cd cale
```

unde *cale* este o cale absolută sau relativă, care precizează catalogul scontat.

### Sugestie

Majoritatea sistemelor UNIX încorporează calea asociată catalogului curent de lucru, sau cel puțin o parte a acesteia, ca parte a promptului de comandă. Dacă sistemul dumneavoastră nu procedează astfel, solicitați-l pe administratorul sistemului pentru a vă ajuta să-l configurați astfel încât să prezinte această caracteristică. Astfel, vă va fi mai simplu să cunoașteți în permanență identitatea catalogului curent de lucru.

### Vizualizarea conținutului catalogului

Pentru a vizualiza numele fișierelor și ale cataloagelor stocate în catalogul curent de lucru, emiteți comanda

```
ls
```

Sau, dacă doriți să vizualizați numele fișierelor și ale cataloagelor incluse într-un alt catalog, emiteți comanda

```
ls cale
```

unde *cale* este o cale absolută sau relativă care precizează catalogul.

Pentru a vizualiza, alături de numele fișierelor și ale cataloagelor, și caracteristicile acestora, adăugați indicatorul *-l* la comanda *ls*:

```
ls -l
```

sau

```
ls -l cale
```

În mod prestabilit, comanda *ls* nu afișează cataloagele sau fișierele al căror nume începe cu un punct; se spune că asemenea fișiere și cataloage sunt *ascunse*. Pentru a vizualiza fișierele și cataloagele ascunse și omoloagele lor vizibile, adăugați indicatorul *-a* la comanda *ls*:

```
ls -a
```

sau

```
ls -al cale
```

### Notă

Comanda *ls* și alte comenzi care manipulează cataloage vor eșua dacă utilizatorul nu are privilegii adecvate pentru accesul la catalog. Vezi sub-sectiunea intitulată „Privilegii de catalog”.

## Crearea unui catalog

Pentru a crea un catalog, emiteți comanda `mkdir`:

```
mkdir cale
```

unde *cale* este o cale absolută sau relativă care precizează catalogul ce urmează a fi creat.

## Ștergerea unui catalog

Pentru a șterge un catalog, emiteți comanda `rmdir`:

```
rmdir cale
```

unde *cale* este o cale absolută sau relativă, care precizează catalogul ce urmează a fi șters. Catalogul trebuie să fie vid; în caz contrar, comanda va eșua.

## Privilegiile de catalog

Ca și fișierele, cataloagele au privilegii asociate. Privilegiile unui catalog se notează folosind aceleași litere care indică privilegiile fișierelor; cu toate acestea, literele au semnificații oarecum diferite atunci când se aplică asupra cataloagelor:

- *r*, catalogul poate fi citit; cu alte cuvinte, sub-cataloagele și fișierele pe care le conține pot fi afișate prin intermediul comenzii `ls` și a altor mijloace similare
- *w*, catalogul poate fi scris; cu alte cuvinte, sub-cataloagele și fișierele pe care le conține pot fi create în catalog și apoi șterse de acolo
- *x*, catalogul se poate folosi; cu alte cuvinte, sub-cataloagele și fișierele pe care le conține sunt accesibile



### Sfatul specialistului

**Întrebare:** Ștergerea unui catalog care conține mai multe sub-cataloage și fișiere este o operație greoaie. Există vreo modalitate mai simplă?

**Răspuns:** Pentru a șterge un catalog și întreg conținutul său, emiteți comanda `rm -rf cale`

unde *cale* este o cale absolută sau relativă, care precizează catalogul ce urmează a fi șters. Procedați cu mare atenție atunci când folosiți această comandă, deoarece, în general, cataloagele și fișierele șterse nu pot fi recuperate.

Când este executată cu argumentul `-1`, comanda `ls` indică privilegiile asociate unui catalog. Utilizatorii cu privilegii adecvate pot modifica privilegiile asociate unui catalog. Pentru aceasta, folosiți comanda `chmod`, care a fost descrisă anterior în acest modul.



### Test „la minut”

- Care este comanda UNIX ce se poate folosi pentru a vizualiza conținutul unui fișier?
- Care este comanda UNIX ce se poate folosi pentru ștergerea unui fișier?
- Care sunt operațiile permise posesorului unui fișier de către setul de privilegii `r-xr-----`?
- Care este comanda UNIX care precizează catalogul curent de lucru?

## Lucrul cu fișiere

Această secțiune se bazează pe noțiunile fundamentale însușite în secțiunea anterioară, prezentând modul de lucru cu fișiere prin utilizarea limbajului PHP. Deoarece modelul sistemului de fișiere folosit în Microsoft Windows diferă de modelul sistemului de fișiere folosit în UNIX, unele funcții PHP nu funcționează corect sub Windows. De asemenea, pentru a complica și mai mult lucrurile, versiunea PHP 4.02 a introdus unele modificări în ceea ce privește modul de lucru al limbajului PHP cu fișierele și cataloagele. Deci, secțiunea de față tratează PHP versiunea 4.02 și versiunile ulterioare sub UNIX. Expunerea indică incompatibilitățile majore, dar trebuie să știți că, dacă folosiți Windows sau o versiune anterioară a limbajului PHP, veți descoperi că unele exemple nu funcționează așa cum ar trebui.

## Aspecte legate de proprietate și privilegii

Deși numeroși programatori PHP folosesc fișiere pentru stocarea datelor, trebuie să vă gândiți bine înainte de a lua această decizie. Stocarea datelor în fișiere poate duce la afectarea caracterului privat al informațiilor pe care le conțin, respectiv la alterarea sau chiar distrugerea fișierelor.

Să ne reamintim că privilegiile asociate unui fișier sau unui catalog determină operațiile pe care le poate executa un utilizator cu fișierul sau catalogul respectiv. Când PHP rulează, o face sub un cont de utilizator desemnat. Privilegiile asociate fișierului sau catalogului determină operațiile pe care PHP le poate executa.

Un administrator de sistem poate configura contul de utilizator sub care rulează PHP. Deseori, administratorii de sistem configurează PHP astfel încât să ruleze sub un cont special de utilizator, care are privilegii foarte limitate; frecvent, acest cont se numește *nobody* (în traducere *nimeni*). Rularea PHP sub un cont cu privilegii limitate

Răspunsuri la test:

- `more`
- `rm`
- `read, execute`
- `pwd`

ii ajută pe administratori să protejeze PHP împotriva pericolelor la adresa securității și este considerat un obicei bun.

Cu toate acestea, dacă administratorul dumneavoastră de sistem a configurat PHP astfel încât să ruleze sub un cont precum nobody, a pune fișierele dumneavoastră la dispoziția PHP devine o problemă. O modalitate constă în a permite tuturor utilizatorilor să aibă acces la fișiere. Dacă doriți ca PHP să poată citi fișierele, această metodă este acceptabilă. Dar, dacă fișierele conțin informații confidențiale sau dacă doriți ca PHP să poată scrie în fișiere, metoda nu este recomandată. Utilizarea acesteia poate duce la un acces neautorizat la informațiile confidențiale incluse în fișiere, respectiv la alterarea sau chiar la distrugerea fișierelor.

O altă metodă constă în a solicita administratorului dumneavoastră de sistem să modifice proprietatea asupra fișierelor la care doriți ca PHP să aibă acces. De exemplu, administratorul de sistem poate repartiza fișierele grupului nobody. Dumneavoastră, ca proprietar al fișierelor, puteți beneficia de privilegii complete, iar PHP poate avea privilegiile disponibile pentru membrii grupului. Totuși, această metodă permite oricărui utilizator să scrie un script PHP și să obțină acces la fișiere cu aceleași privilegii ca și cele permise PHP. În general, și această metodă se va dovedi nesatisfăcătoare.

Cu excepția cazurilor când sunteți administrator de sistem și când nici un utilizator neautorizat nu poate obține acces la sistem, probabil că nu veți reuși să concepeți o metodă care să vă permită să folosiți fișiere pentru a stoca într-o manieră sigură date confidențiale sau volatile fără a risca o compromitere a fișierelor sau chiar a sistemului însuși. O modalitate mai sigură de stocare a datelor confidențiale sau volatile constă în utilizarea unei baze de date. În general, fișierele constituie un mijloc adecvat numai pentru stocarea datelor publice, non-volatile, care vor fi accesibile sistemului PHP.

## Obținerea atributelor unui fișier

PHP furnizează numeroase funcții care vă permit să obțineți informații care descriu un fișier. Tabelul 11-2 rezumă cele mai cunoscute dintre aceste funcții.

Funcțiile `fileowner()` și `filegroup()` returnează fiecare un identificator numeric; puteți converti identificatorul numeric într-un șir prin invocarea funcției `posix_getpwuid()` cu un identificator de utilizator, respectiv a funcției `posix_getgrgid()` cu un identificator de grup.

**Tabelul 11-2** Funcții PHP pentru obținerea atributelor unui fișier

Funcție	Descriere
<code>file_exists()</code>	Returnează <code>true</code> dacă fișierul specificat există, respectiv <code>false</code> în caz contrar.

Funcție	Descriere
<code>filetime()</code>	Returnează timpul de acces la fișier sub formă de amprentă de timp UNIX.
<code>filectime()</code>	Returnează timpul de modificare al i-nodului (structură de date care conține informații despre fișiere UNIX – N. T.) sub formă de amprentă de timp UNIX.
<code>filegroup()</code>	Returnează identificatorul numeric al grupului care deține fișierul.
<code>filemtime()</code>	Returnează momentul de timp al modificării fișierului sub formă de amprentă de timp UNIX.
<code>fileowner()</code>	Returnează identificatorul numeric de utilizator al fișierului.
<code>fileperms()</code>	Returnează permisiunile fișierului.
<code>filesize()</code>	Returnează dimensiunea fișierului, în octeți.
<code>filetype()</code>	Returnează tipul fișierului, în speță "fifo", "char", "dir", "block", "link", "file" sau "unknown".
<code>is_dir()</code>	Returnează <code>true</code> dacă fișierul specificat există și este un catalog; în caz contrar, returnează <code>false</code> .
<code>is_file()</code>	Returnează <code>true</code> dacă fișierul specificat există și este un fișier obișnuit; în caz contrar, returnează <code>false</code> .
<code>is_readable()</code>	Returnează <code>true</code> dacă fișierul specificat există și poate fi citit; în caz contrar, returnează <code>false</code> .
<code>is_writable()</code>	Returnează <code>true</code> dacă fișierul specificat există și se poate scrie în acel fișier; în caz contrar, returnează <code>false</code> .

Un exemplu care indică modul de invocare a acestor funcții este rezident în situl Web al cărții (<http://www.osborne.com>), astfel încât să puteți descărca și rula personal fișierul. Numele fișierului este `attributes.php`. Iată conținutul fișierului:

```
<?php
$filename = "test.txt";

$result = file_exists($filename);
echo "
file_exists(): $result";

$result = posix_getpwuid(fileowner($filename));
$result = $result["name"];
echo "
fileowner(): $result";

$result = posix_getgrgid(filegroup($filename));
$result = $result["name"];
echo "
filegroup(): $result";

$result = filetype($filename);
echo "
filetype(): $result";

$result = filesize($filename);
echo "
filesize(): $result";

$result = filetime($filename);
$result = date("m/d/Y H:i", $result);
echo "
filetime(): $result";

$result = filectime($filename);
$result = date("m/d/Y H:i", $result);
echo "
filectime(): $result";
```

```

$result = filetime($filename);
$result = date("m/d/Y H:i", $result);
echo "
filetime(): $result";

$result = fileperms($filename);
$result = decoct($result);
echo "
fileperms(): $result";

$result = is_file($filename);
echo "
is_file(): $result";

$result = is_dir($filename);
echo "
is_dir(): $result";

$result = is_readable($filename);
echo "
is_readable(): $result";

$result = is_writable($filename);
echo "
is_writable(): $result";
?>

```

Dacă rulați personal exemplul prezentat anterior, nu uitați să modificați valoarea variabilei `$filename` (nume fișier), folosind numele unui fișier rezident în același catalog ca și scriptul. Datele de ieșire ale exemplului sunt asemănătoare celor prezentate în listingul următor:

```

file_exists(): 1
fileowner(): nobody
filegroup(): nobody
filetype(): file
filesize(): 0
fileatime(): 04/12/2005 10:04
filectime(): 04/11/2005 11:28
filemtime(): 04/10/2005 10:36
fileperms(): 100644
is_file(): 1
is_dir(): 0
is_readable(): 1
is_writable(): 0

```

## Modificarea privilegiilor unui fișier

Pentru a modifica privilegiile unui fișier, invocați `chmod()`, o funcție ale cărei argumente sunt asemănătoare celor folosite în linia de comandă UNIX:

```
chmod(nume_fisier, mod);
```

Argumentul *nume\_fisier* specifică numele sau calea de acces a fișierului ale cărui privilegii urmează a fi modificate, iar argumentul *mod* specifică privilegiile dorite. În general, se preferă exprimarea privilegiilor sub forma unui număr scris în octal. Pentru aceasta, prefixați valoarea folosind cifra 0. De exemplu, valoarea literală 010 are valoarea 8, nu valoarea 10. Ca atare, puteți atribui unui fișier privilegiile `rw-r-----` specificând valoarea 0640.

Pentru ca funcția `chmod()` să se execute cu succes, PHP trebuie să ruleze sub contul utilizatorului posesor al fișierului. Funcția returnează `true` în caz de reușită, respectiv `false` în caz contrar.

### Atenție

Funcția `chmod()` nu funcționează sub Microsoft Windows.

## Modificarea proprietății asupra unui fișier

Pentru a modifica grupul posesor al unui fișier, invocați funcția `chgrp()`, care are forma:

```
chgrp(nume_fisier, grup);
```

unde *nume\_fisier* este numele sau calea fișierului, iar *grup* este numele sau identificatorul numeric al grupului. Pentru ca funcția să se execute cu succes, contul de utilizator sub care rulează PHP trebuie să fie posesor al fișierului și să fie membru al grupului specificat prin argumentul *grup*. Funcția returnează `true` în caz de reușită, respectiv `false` în caz contrar.

### Atenție

Funcția `chgrp()` nu funcționează sub Microsoft Windows.

### Notă

Deși PHP include o funcție `chown()` care modifică proprietatea asupra unui fișier, PHP trebuie să ruleze folosind contul rădăcină pentru ca această funcție să poată da rezultate. Deoarece acesta este un procedeu nesigur și rareori folosit, în general funcția `chown()` nu este disponibilă.

## Deschiderea unui fișier

Procesul de stabilire a accesului la un fișier se numește *deschiderea* fișierului. Înainte de a putea citi sau scrie într-un fișier, trebuie să deschideți fișierul folosind funcția `fopen()`:

```
fopen(nume_fisier, mod);
```

unde *nume\_fisier* specifică numele sau calea spre fișierul care urmează a fi deschis, iar *mod* indică tipul de acces dorit. De exemplu, instrucțiunea următoare deschide fișierul `o_carte.txt` pentru citire:

```
$fh = fopen("o_carte.txt", "r");
```

Observați că funcția `fopen()` returnează o valoare. Această valoare este `false` dacă PHP nu a reușit să deschidă fișierul. În caz contrar, această valoare conține un

întreg care se numește *identificator de fișier*, care se folosește pentru identificarea unui fișier de către funcțiile care execută operații cu fișiere. Uneori, această valoare se numește *pointer de fișier*. Cu toate acestea, termenul *identificator de fișier* este mai exact, deoarece noțiunea de *identificator* face referire în mod corespunzător la o valoare întreagă, în timp ce termenul de *pointer* face referire la o adresă din memorie.

Tabelul 11-3 prezintă valorile argumentului `mod` care pot fi transmise funcției `fopen()`. Literele care desemnează modurile corespund cuvintelor *read* (a citi), *write* (a scrie) și *append* (a atașa). Un mod ("r") permite accesul la citire. Două moduri permit accesul la scriere (acestea sunt "w" și "a"). Trei moduri (cele care includ un semn +) permit ambele tipuri de acces. Unele moduri determină PHP să încerce a crea fișierul, dacă acesta nu există. Două moduri trunchiază un fișier existent, adică șterg conținutul fișierului, nu și fișierul în sine.

### Atenție

Pentru ca funcția `fopen()` să se execute cu succes, PHP trebuie să ruleze sub contul unui utilizator care dispune de suficiente privilegii pentru a executa operațiile indicate de modul respectiv. De exemplu, pentru a putea crea un fișier, PHP trebuie să ruleze ca utilizator cu acces de scriere la catalogul în care urmează a fi creat fișierul.

Fiecare fișier are un pointer asociat, care indică amplasarea octetului din fișier unde se va produce următoarea operație (citire sau scriere). Valoarea modului funcției `fopen()` determină valoarea inițială a pointerului de fișier. Secțiunea următoare, „Navigarea într-un fișier”, explică modul de acces la pointerul de fișier și modul de manipulare a acestuia.

PHP furnizează o formă alternativă a funcției `fopen()`, care preia trei argumente:

```
fopen(ume_fisier, mod, cale)
```

**Tabelul 11-3** Moduri folosite cu funcția `fopen()`

Mod	Citire	Scriere	Creare	Trunchiere	Pointer
"r"	x				Început
"r+"	x	x			Început
"w"		x	x	x	Început
"w+"	x	x	x	x	Început
"a"		x	x		Sfârșit
"a+"	x	x	x		Sfârșit

Dacă argumentul `cale` are valoarea "1", PHP va căuta fișierul într-un catalog special, denumit *cale de includere*. Administratorul PHP configurează identitatea căii de includere. Dacă specificați argumentul `cale`, argumentul `ume_fisier` trebuie să fie alcătuit dintr-un nume de fișier sau o cale relativă, nu dintr-o cale absolută.

### Atenție

În mod prestabilit, PHP 4.04p1 raportează că este configurată calea de includere. Cu toate acestea, este necesară stabilirea manuală a valorii variabilei `include_path` din fișierul `php.ini`; în caz contrar, calea de includere nu va fi folosită.

### Atenție

Sub Microsoft Windows, fișierele ASCII și fișierele binare sunt tratate în mod diferit. Când deschideți un fișier binar sub Windows, specificați b ca al doilea caracter al modului; de exemplu, "rb". Dacă nu procedați astfel, citirile din fișier și alte operații se vor încheia prematur sau vor eșua.

## Verificarea finalizării unei operații cu un fișier

Operațiile cu fișiere – inclusiv cele legate de deschiderea, citirea unui fișier și scrierea într-un fișier – pot eșua dintr-o varietate de motive. Deci, este important să verificați dacă fiecare operație s-a încheiat cu succes. Iată o modalitate în care puteți proceda:

```
$fh = fopen("o_carte.txt", "r");
if (!$fh)
{
 die("Nu a fost deschis fisierul <I>o_carte.txt</I>.");
}
```

Funcția `fopen()` returnează false dacă nu reușește să deschidă fișierul. În acest caz, scriptul invocă funcția `die()` pentru a afișa un mesaj și pentru a-și încheia execuția.

Cu mult mai compacte, dar și posibil mai derutante, sunt formele folosite de programatorii PHP cu experiență. De exemplu:

```
($fh = fopen("o_carte.txt", "r"))
|| die("Nu a fost deschis fisierul <I>o_carte.txt</I>.");
```

Parantezele și utilizarea caracterelor spațiu alb contribuie la clarificarea acestei instrucțiuni. Instrucțiunea invocă funcția `fopen()` și atribuie rezultatul variabilei `$fh`. Apoi, execută o operație SAU logic (simbolizată prin operatorul `||`) cu doi operanzi, în speță `$fh` și `die()`. Dacă variabila `$fh` are valoarea true, rezultatul operației SAU logic va fi întotdeauna true; deci, în acest caz, PHP nu execută evaluarea celuilalt operand al operației SAU logic, adică invocarea funcției `die()`. Ca atare, funcția `die()` este executată numai dacă variabila `$fh` are valoarea false, adică dacă funcția `fopen()` eșuează.

O formă alternativă mai simplă folosește operatorul OR (SAU):

```
$fh = fopen("o_carte.txt", "r")
OR die("Nu a fost deschis fisierul <I>o_carte.txt</I>.");
```

Această formă are același principiu de funcționare ca și forma anterioară. Totuși, operatorul SAU are o precedență mai redusă decât operatorul ||, deci este posibilă scrierea instrucțiunii cu un număr mai redus de paranteze.

### Sugestie

Să ne reamintim că prin prefixarea, cu ajutorul caracterului @, a numelui unei funcții invocate, limbajul PHP elimină avertismentele și alte mesaje generate în timpul execuției funcției. Dacă testați cu sârguință rezultatul fiecărei operații cu fișiere, puteți elimina cu ușurință mesajele de eroare și implicit evitați amestecarea datelor de ieșire ale scriptului dumneavoastră cu mesajele PHP, care în caz contrar pot deruta utilizatorul.

## Închiderea unui fișier

Un fișier deschis consumă resursele sistemului. Când un script a terminat de utilizat un fișier, scriptul trebuie să *închidă* fișierul, eliberând aceste resurse. La sfârșitul unui script, PHP închide în mod automat fișierele deschise. Totuși, la programare se recomandă să închideți fișierele mai rapid, ori de câte ori este posibil.

Pentru a închide un fișier, invocați funcția `fclose()`:

```
fclose(identificator_fisier)
```

unde `identificator_fisier` este identificatorul fișierului, returnat la deschiderea acestuia. De exemplu, iată un exemplu caracteristic de deschidere, utilizare și închidere a unui fișier:

```
$fh = @fopen("o_carte.txt", "r");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>o_carte.txt</I>.");
}
// Aici se insereaza instructiunile care folosesc fișierul deschis
@fclose($fh);
```

### Sugestie

Funcția `fclose()` returnează valoarea `true()` dacă fișierul a fost închis cu succes. Testarea acestei valori este rareori necesară, deoarece nu se mai pot face prea multe după ce s-a ratat o încercare de a închide un fișier.

## Citirea dintr-un fișier

PHP furnizează o varietate de funcții pentru citirea fișierelor. Prima dintre acestea este `fread()`, care are următoarea formă:

```
fread(identificator_fisier, lungime)
```

Argumentul `identificator_fisier` este valoarea returnată de funcția `fopen()`, iar argumentul `lungime` specifică numărul maxim de octeți care vor fi citați. Octeții citați

din fișier sunt returnați sub formă de valoare de tip șir. Dacă operația de citire întâlnește sfârșitul fișierului, PHP va returna mai puțin de *lungime* octeți.

### Sugestie

PHP include o funcție conexă, denumită `fgetc()`, care citește un octet din fișierul specificat.

Iată un exemplu care prezintă modul de citire și de afișare a unui text dintr-un fișier:

```
$fh = @fopen("o_carte.txt", "r");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>o_carte.txt</I>.");
}
$ss = fread($fh, 256);
echo "
Citeste: $s";
fclose($fh);
```

Exemplul citește maximum 256 de octeți din fișier. Pentru a citi un număr mai mare sau mai mic de octeți, modificați valoarea argumentului *lungime* al funcției `fread()`.

### Sugestie

Dacă doriți să încercați personal acest exemplu, asigurați-vă că fișierul `o_carte.txt` se află în același catalog ca și scriptul dumneavoastră și că PHP are permisiunea de a citi fișierul.

Pentru a citi și a afișa întregul conținut al fișierului, folosiți funcția `filesize()` pentru a furniza valoarea argumentului *lungime*, astfel:

```
$nume_fisier = "o_carte.txt";
$fsh = @fopen($nume_fisier, "r");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>$nume_fisier</I>.");
}
$ss = fread($fh, filesize($nume_fisier));
echo "
Citeste: $s";
fclose($fh);
```

## Citirea unei linii de text

O linie de text este o serie de caractere urmate de un caracter de terminare a liniei. Se obișnuiește frecvent citirea linie cu linie a unui text dintr-un fișier. Funcția `fgets()` citește o linie dintr-un fișier; funcția are următoarea formă:

```
fgets(identificator_fisier, lungime)
```

Ca și în cazul funcției `fread()`, argumentul `identificator_fisier` este o valoare returnată de funcția `fopen()`; cu toate acestea, argumentul *lungime* specifică numărul



maxim de octeți care vor fi citiți, *minus o unitate*, pentru a permite includerea caracterului de terminare a liniei. Octeții citiți din fișier sunt returnați ca valoare de tip șir.

Iată un exemplu care prezintă modul de citire și de afișare a primei linii a unui fișier:

```
$fh = @fopen("o_carte.txt", "r");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>o_carte.txt</I>.");
}
$s = fgets($fh, 256);
echo "
Citeste: $s";
fclose($fh);
```

În exemplul anterior s-a presupus că linia cea mai lungă a fișierului conține mai puțin de 256 octeți. Pentru a permite lungimi de linie mai mari, pur și simplu modificați valoarea argumentului *lungime* al funcției `fgets()`.

### Citirea linie cu linie a unui întreg fișier

În general, dintr-un fișier trebuie citit mai mult decât prima linie a acestuia. Pentru aceasta, un script trebuie să dispună de o modalitate de a determina momentul când fișierul a fost citit în totalitate. Funcția `feof()` are chiar acest scop, returnând o valoare care arată dacă s-a ajuns sau nu la sfârșitul fișierului. Funcția are următoarea formă:

```
feof(identificator_fisier)
```

Argumentul *identificator\_fisier* este valoarea returnată de funcția `fopen()`. Funcția `feof()` returnează `true` dacă fișierul specificat este la sfârșit; în caz contrar, returnează `false`.

Iată cum se poate folosi funcția `feof()` pentru a controla procesul de citire a unui întreg fișier, linie cu linie:

```
$fh = @fopen("o_carte.txt", "r");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>o_carte.txt</I>.");
}
$s = fgets($fh, 256);
while (!feof($fh))
{
 echo "
Citeste: $s";
 $s = fgets($fh, 256);
}
fclose($fh);
```

Instrucțiunea `while` asigură faptul că funcția `fgets()` este apelată în mod repetat, până la citirea tuturor liniilor.

O modalitate mai simplă de a citi linie cu linie un întreg fișier constă în a folosi funcția `file()`. Această funcție returnează un tablou în care fiecare element conține

o linie a fișierului text specificat. Iată un exemplu care folosește funcția `file()` pentru a citi și pentru a afișa conținutul unui fișier:

```
$tablou = file("o_carte.txt");
foreach ($tablou as $s)
{
 echo "
Citeste: $s";
}
```

### Atenție

Această metodă nu este adecvată pentru fișiere de foarte mari dimensiuni, deoarece în tablou este încărcat întregul conținut al fișierului, ceea ce poate necesita o cantitate de memorie superioară celei disponibile.

### Afișarea conținutului unui fișier

PHP furnizează două funcții care facilitează afișarea conținutului unui fișier. Una dintre funcții, `fpassthru()`, necesită un argument care specifică identificatorul fișierului care urmează să fie afișat:

```
$fh = fopen("o_carte.txt", "r");
fpassthru($fh);
```

După ce a afișat fișierul, funcția îl închide automat.

Cealaltă funcție, `readfile()`, necesită numai numele sau calea fișierului:

```
readfile("o_carte.txt");
```

### Navigarea printr-un fișier

Așa cum s-a arătat anterior, fiecare fișier are un pointer asociat care indică poziția octetului unde se va produce următoarea operație. Puteți folosi funcția `rewind()` pentru a readuce pointerul la începutul fișierului. Funcția are următoarea formă:

```
rewind(identificator_fisier)
```

unde *identificator\_fisier* este identificatorul de fișier returnat de funcția `fopen()`.

### Notă

Nu puteți readuce pointerul unui fișier la începutul propriu-zis al unui fișier dacă fișierul a fost deschis pentru un acces de tip atașare, adică într-unul din modurile `'a'` sau `'a+'`.

Iată un exemplu care prezintă modul de utilizare a funcției `rewind()` pentru a afișa de două ori conținutul unui fișier:

```
$fh = @fopen("o_carte.txt", "r");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>o_carte.txt</I>.");
}
```

```

}
$s = fgets($fh, 256);
while (!feof($fh))
{
 echo "
\$s: \$s";
 $s = fgets($fh, 256);
}
// Deruleaza la inceputul fisierului si reia redarea acestuia
rewind($fh);
$s = fgets($fh, 256);
while (!feof($fh))
{
 echo "
\$s: \$s";
 $s = fgets($fh, 256);
}
fclose($fh);

```

Dacă se produce vreo eroare, funcția `rewind()` returnează zero.

Deși funcția `rewind()` este utilă în caz de nevoie, necesitatea de a readuce un pointer de fișier la începutul fișierului nu este chiar atât de frecventă. Funcția `fseek()` furnizează o mai mare flexibilitate, permițându-vă să poziționați pointerul de fișier astfel încât să puteți citi sau scrie în orice punct al fișierului. Funcția are două forme, cea mai simplă fiind următoarea:

```
fseek(identificator_fisier, offset)
```

unde *identificator\_fisier* este identificatorul de fișier returnat de funcția `fopen()`, iar *offset* este poziția dorită a pointerului de fișier, specificată în octeți, în raport cu începutul fișierului. În caz de reușită, funcția `fseek()` returnează 0; în caz contrar, returnează -1.

O formă alternativă a funcției asigură un grad superior de flexibilitate:

```
fseek(identificator_fisier, offset, baza)
```

unde *identificator\_fisier* și *offset* au semnificațiile definite anterior, iar *baza* ia una dintre următoarele valori:

- `SEEK_SET`, care stabilește poziția pointerului de fișier în raport cu începutul fișierului
- `SEEK_CUR`, care stabilește poziția pointerului de fișier în raport cu valoarea curentă a pointerului
- `SEEK_END`, care stabilește poziția pointerului de fișier relativ la sfârșitul fișierului

Valoarea argumentului *offset* poate fi pozitivă, negativă sau zero.

De exemplu, pentru a poziționa pointerul cu 1000 de octeți înainte de sfârșitul fișierului, emiteți următorul apel de funcție:

```
fseek($fh, -1000, SEEK_END)
```

unde *\$fh* este identificatorul fișierului al cărui pointer doriți să-l repoziționați.

Pentru a obține valoarea curentă a pointerului de fișier, invocați funcția `ftell()`, care are următoarea formă:

```
ftell(identificator_fisier)
```

Funcția returnează valoarea curentă a identificatorului de fișier, respectiv valoarea zero dacă funcția eșuează.

## Scrierea într-un fișier

Spre deosebire de varietatea de funcții furnizate pentru citirea fișierelor, PHP oferă o singură funcție pentru scrierea în fișiere, și anume `fwrite()`. Funcția are următoarea formă:

```
fwrite(identificator_fisier, data)
```

unde *identificator\_fisier* este identificatorul de fișier returnat de funcția `fopen()`, iar *data* este o valoare șir care determină datele care urmează a fi scrise. Dacă execuția funcției reușește, returnează numărul octeților scriși; în caz contrar, returnează valoarea -1.

Iată un exemplu care prezintă modul de scriere a datelor într-un fișier:

```

$fh = @fopen("jurnal.txt", "a");
if (!$fh)
{
 die("Nu a fost deschis fisierul <I>jurnal.txt</I>.");
}
$ok = fwrite($fh, "Acestea sunt date bune.\n");
echo "
Rezultatul scrierii: $ok";
fclose($fh);

```

Programul prezentat în exemplul anterior scrie în fișier o linie de text. Deoarece fișierul a fost deschis folosind modul "a", datele sunt atașate la fișier; cu alte cuvinte, datele sunt scrise *după* toate datele existente în fișier. Observați că a fost scris și un caracter de terminare a liniei ("  
"), astfel încât fișierul să poată fi citit linie cu linie la un moment de timp ulterior. Dacă lucrați cu un fișier text, în general trebuie să includeți un caracter de terminare a liniei la sfârșitul fiecărei linii scrise în fișier.

Programul din exemplu scrie în fișier o singură linie de text. Totuși, puteți scrie mai multe linii, dacă doriți. Dacă apelați funcția `fwrite()` din interiorul unei bucle, aveți posibilitatea de a scrie mai multe linii. După ce ați scris toate liniile de care aveți nevoie, închideți fișierul prin apelarea funcției `fclose()`.

PHP mai furnizează și o formă alternativă a funcției `fwrite()`:

```
fwrite(identificator_fisier, data, lungime)
```

Această formă include un al treilea argument, și anume *lungime*, care vă permite să specificați numărul maxim de octeți care vor fi scriși.

## Sugestie

Argumentul lungime al funcției `fwrite()` trebuie utilizat atunci când se scrie în fișiere binare sub Microsoft Windows. De asemenea, Windows preferă secvența de terminare a liniei `"\r\n"`. Atunci când scrieți programe PHP pentru sisteme Windows, este de preferat să folosiți secvența Windows de terminare a liniei.

## Sugestie

PHP dispune de o altă funcție pentru scrierea fișierelor, în speță `fputs()`. Totuși, în afară de nume, `fputs()` este una și aceeași funcție cu `fwrite()`. Deci, practic, `fputs()` și `fwrite()` reprezintă una și aceeași funcție, cu nume diferite.



### Proiect 11-1: Un contor pentru numărul de deschideri ale unei pagini

În cadrul acestui proiect, veți construi un script care numără de câte ori s-a obținut accesul la o anumită pagină Web.

#### Scopurile proiectului

- Prezentarea modului de citire și scriere a unui fișier folosind PHP
- Prezentarea modului de creare a unui contor utilizat într-o pagină Web

#### Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit `ctr.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
$file = basename($PHP_SELF) . ".dat";
$fh = @fopen($file, "r+");
if (!$fh)
{
 die("Nu a fost deschis fișierul <I>$file</I>.");
}
$s = fgetc($fh, 6);
$count = (int) $s;
$count = $count + 1;
$count = str_pad($count, 6);
rewind($fh);
fwrite($fh, $count);
echo " $count ";
fclose($fh);
?>
```

2. Plasați următorul script PHP într-un fișier numit `ctr-test.php` și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul `ctr.php`:

```
<HTML>
<HEAD>
<TITLE>ctr-test.php</TITLE>
</HEAD>
<BODY>
Aceasta pagina a fost deschisă de
```

```
<?php include "ctr.php" ?>
 ori.
</BODY>
</HTML>
```

3. Creați fișierul `ctr-test.php.dat` în același catalog care conține scripturile pe care le-ați încărcat. Puteți crea fișierul prin emiterea comenzii

```
touch ctr-test.php.dat
```

Asigurați-vă că PHP are acces de citire și de scriere la fișier. Dacă nu știți precis cum trebuie să efectuați această operație, luați legătura cu administratorul dumneavoastră de sistem.

4. Alocăți un timp studiului scriptului PHP `ctr.php`. Observați modul în care scriptul folosește variabila PHP `$PHP_SELF` pentru a determina numele fișierului care o conține și apoi atașează particula `.dat` pentru a forma numele fișierului care conține contorul de pagină. De asemenea, remarcați funcțiile pe care le folosește scriptul pentru a executa operațiile cu fișiere.
5. Alocăți un timp studiului scriptului PHP `ctr-test.php`. Remarcați că acest script este alcătuit mai ales din coduri HTML; în fișier apare o singură linie PHP. Instrucțiunea `include` determină limbajul PHP să includă conținutul fișierului `ctr.php` în fișierul `ctr-test.php` atunci când este deschis acesta din urmă.
6. Orientați un browser Web spre adresa URL a scriptului `ctr-test.php`. Browserul trebuie să afișeze numărul de deschideri ale paginii. Prin reîmprospătarea paginii se determină incrementarea contorului din pagină.

## Obținerea accesului exclusiv la un fișier

Web-ul ridică o problemă specială pentru dezvoltatorii de programe, deoarece mai mulți utilizatori pot avea acces simultan la un singur script PHP. Uneori, accesul simultan poate avea ca rezultat deteriorarea fișierului. Pentru a vedea cum se poate produce aceasta, să luăm în considerare procesul de acces și actualizare a contorului de pagini asociat unei pagini Web. Să presupunem că, inițial, contorul paginii are valoarea 100. Acest proces implică următoarea succesiune de evenimente:

1. Deschide fișierul care conține valoarea contorului.
2. Citește fișierul, obținând valoarea curentă a contorului (100).
3. Incrementează contorul (101).
4. Scrie în fișier, stocând valoarea actualizată a contorului (101).
5. Închide fișierul.

Acum, să vedem ce se poate întâmpla dacă două procese obțin acces simultan la pagina Web și la contorul său, astfel încât etapele celor două procese se întrepătrund. Tabelul 11-4 prezintă rezultatul.

Tabelul 11-4 Acces conflictual la un fișier

Procesul nr. 1	Procesul nr. 2
1. Deschide fișierul care conține valoarea contorului.	1. Deschide fișierul care conține valoarea contorului.
2. Citește fișierul, obținând valoarea curentă a contorului (100).	2. Citește fișierul, obținând valoarea curentă a contorului (100).
3. Incrementează contorul (101).	3. Incrementează contorul (101).
4. Scrie în fișier, stocând valoarea actualizată a contorului (101).	4. Scrie în fișier, stocând valoarea actualizată a contorului (101).
5. Închide fișierul.	5. Închide fișierul.

Observați că procesul 2 citește fișierul înainte ca procesul 1 să scrie valoarea actualizată a contorului. Ca atare, ambele procese obțin valoarea 100 ca valoare inițială a contorului, incrementează contorul la 101 și scriu 101 ca valoare incrementată a contorului. Deci, deși două procese au obținut accesul la pagină, contorul paginii s-a incrementat cu numai o unitate. În cazul în care compensația dumneavoastră depinde de numărul vizitatorilor paginii, aceasta este o problemă extrem de serioasă.

Pentru a evita problema, este necesar ca fiecare proces să obțină acces exclusiv la fișierul care conține contorul din pagină. Astfel, accesul la fișier este succesiv: fișierul este folosit mai întâi de un proces, și apoi de celălalt. Atâta vreme cât procesele sunt împiedicate să obțină simultan accesul la fișier, contorul nu va indica valori eronate.

Funcția `flock()` permite unui proces să obțină un acces exclusiv la un fișier, acordând procesului o blocare a fișierului. Funcția are forma:

```
flock(identificator_fișier, operație)
```

unde *identificator\_fișier* este identificatorul de fișier returnat de funcția `fopen()`. Argumentul *operație* este o constantă sau o expresie care specifică operația dorită, după cum urmează:

Operație	Descriere
LOCK_SH	Obține o blocare partajată a fișierului specificat. Mai multe procese pot deține o blocare partajată; pentru un fișier însă, nici un proces nu poate obține o blocare partajată în timp ce un alt proces deține o blocare exclusivă. Procesul va aștepta până când are posibilitatea de a obține blocarea cerută.
LOCK_EX	Obține o blocare exclusivă pentru fișierul specificat. Un singur proces poate deține o blocare exclusivă a unui fișier. Procesul va aștepta până când are posibilitatea de a obține blocarea cerută.
LOCK_SH + LOCK_NB	Încearcă să obțină o blocare partajată a fișierului specificat; totuși, nu așteaptă obținerea blocării, dacă aceasta nu este disponibilă imediat.
LOCK_EX + LOCK_NB	Încearcă să obțină o blocare exclusivă pentru fișierul specificat; totuși, nu așteaptă obținerea blocării, dacă aceasta nu este disponibilă imediat.
LOCK_UN	Eliberează blocarea fișierului specificat.

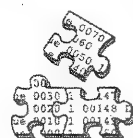
Funcția `flock()` returnează `true` în caz de reușită, respectiv `false` când blocarea solicitată nu poate fi obținută (`LOCK_SH` și `LOCK_EX`) sau eliberată (`LOCK_UN`).

### Atenție

Blocarea este o activitate care se desfășoară în comun. Procesele nu au niciodată accesul interzis la fișiere, ci numai la blocări. Ca atare, un proces care nu încearcă să blocheze un fișier poate obține acces la fișier chiar dacă un alt proces a cerut și a obținut o blocare exclusivă a fișierului.

### Atenție

Unele servere Web, inclusiv IIS de la Microsoft, rulează un singur proces multifir. În asemenea situații, funcția `flock()` nu va reuși să furnizeze acces exclusiv la fișiere.



## Proiect 11-2: Un contor îmbunătățit al accesului la o pagină

În cadrul acestui proiect, veți construi un script care numără de câte ori a fost deschisă o pagină Web. Scriptul de față îl îmbunătățește pe cel prezentat anterior. Scriptul blochează fișierul care conține contorul cu numărul de deschideri ale paginii, astfel încât accesul simultan să nu cauzeze o numărare eronată.

### Scopurile proiectului

- Prezentarea modului de blocare a fișierelor
- Prezentarea modului de implementare a unui contor mai fiabil de cuantificare a accesului la o pagină

### Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit `lctr.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
$fh = @fopen($file, "r+");
 or die("Nu a fost deschis fișierul <I>$file</I>.");
@flock($fh, LOCK_EX)
 or die("Nu poate bloca fișierul <I>$file</I>.");
$ss = @fgets($fh, 6);
$count = (int) $ss + 1;
$count = str_pad($count, 6);
@rewind($fh);
 or die("Nu poate derula fișierul <I>$file</I>.");
if (@fwrite($fh, $count) == -1)
{
 die("Nu poate scrie în fișierul <I>$file</I>.");
}
echo "$count";
@flock($fh, LOCK_UN)
 or die("Nu poate debloca fișierul <I>$file</I>.");
fclose($fh)
 or die("Nu poate închide fișierul <I>$file</I>.");
?>
```

2. Plasați următorul script PHP într-un fișier denumit `lctrtest.php` și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul `lctr.php`:

```
<HTML>
<HEAD>
<TITLE>lctrtest.php</TITLE>
</HEAD>
<BODY>
<?php $file="lctrtest.php.dat" ?>
Aceasta pagina a fost deschisa de
<?php include "lctr.php" ?>
 ori.
</BODY>
</HTML>
```

3. Creați fișierul `lctrtest.php.dat` în același catalog care conține scripturile pe care le-ați încărcat. Puteți crea fișierul prin emiterea comenzii

```
touch lctrtest.php.dat
```

Asigurați-vă că PHP are acces de citire și de scriere la fișier. Dacă nu știți precis care este modul de executare a acestei operații, luați legătura cu administratorul dumneavoastră de sistem.

4. Alocați un timp studiului scriptului PHP `lctr.php`. Remarcați modul de realizare a blocării și a deblocării fișierului.
5. Alocați un timp studiului scriptului PHP `lctrtest.php`. Remarcați că este foarte puțin diferit de scriptul folosit în cadrul Proiectului 11-1.
6. Orientați un browser Web spre adresa URL a scriptului `lctrtest.php`. Browserul va afișa numărul de deschideri ale paginii respective. Prin reîmprospătarea paginii este determinată incrementarea contorului paginii.
7. Orientați două browsere Web spre adresa URL a scriptului `lctrtest.php`. Încercați să determinați browserele să obțină simultan accesul la pagină. Chiar dacă veți reuși această operație, blocările aplicate fișierelor vor asigura actualizarea în mod adecvat a contorului de pagină.

## Copierea unui fișier

PHP furnizează o funcție care facilitează copierea fișierelor, și anume funcția `copy()`. Funcția `copy()` are următoarea formă:

```
copy (sursa, destinație)
```

unde *sursa* este numele sau calea fișierului care urmează a fi copiat, iar *dest* este numele sau calea copiei. Funcția returnează `true` dacă operația de copiere reușește; în caz contrar, returnează `false`.

Iată un exemplu în care este prezentat modul de utilizare a funcției `copy()`:

```
$ok = copy("test.txt", "test.txt.bak");
if (!$ok)
{
 die("
Nu a reusit sa copieze fisierul.");
}
```

Exemplul creează o copie de siguranță a fișierului `test.txt`, cu numele `test.txt.bak`.

## Notă

Rețineți că PHP trebuie să aibă acces de scriere la catalogul în care se va afla copia; în caz contrar, PHP nu va putea crea copia.

## Atenție

Funcția `copy()` va suprascrie fișierul destinație, dacă acesta există.

## Modificarea numelui unui fișier

PHP furnizează o funcție care vă permite să modificați numele unui fișier, și anume `rename()`, funcție care are următoarea formă:

```
rename (nume_vechi, nume_nou)
```

unde *nume\_vechi* este numele sau calea originală a fișierului, iar *nume\_nou* este numele sau calea dorită. Funcția returnează `true` dacă operația de modificare a numelui a reușit; în caz contrar, returnează `false`.

Iată un exemplu care ilustrează modul de utilizare a funcției `rename()`:

```
$ok = rename("test.txt", "TEST.txt");
if (!$ok)
{
 die("
Nu a reusit sa modifice numele fisierului.");
}
```

Exemplul modifică numele fișierului `test.txt`, atribuindu-i numele `TEST.txt`.

## Notă

Rețineți că PHP trebuie să aibă acces de scriere la catalogul în care se va afla fișierul cu numele modificat; în caz contrar, PHP nu va putea modifica numele fișierului.

## Atenție

Ca și în cazul comenzii UNIX `mv`, funcția `rename()` va suprascrie fișierul destinație, dacă acesta există.

## Ștergerea unui fișier

PHP furnizează o funcție care vă permite să ștergeți un fișier, și anume `unlink()`. Funcția are următoarea formă:

```
unlink (nume_fisier)
```

unde *nume\_fisier* este numele sau calea fișierului care urmează a fi șters. Funcția returnează `true` dacă operația de ștergere a avut succes, respectiv `false` în caz contrar.

Iată un exemplu care prezintă modul de utilizare a funcției `unlink()`:

```
$ok = unlink("test.txt");
if (!$ok)
{
 die("
Nu a reusit sa stearga fisierul.");
}
```

Programul din exemplul anterior șterge fișierul `test.txt`.

### Notă

Rețineți că PHP trebuie să aibă acces de scriere la catalogul în care se află fișierul; în caz contrar, PHP nu va putea șterge fișierul.

### Notă

În conformitate cu manualul PHP, funcția `unlink()` „s-ar putea să nu funcționeze” în sistemele Microsoft Windows.

## ? Sfatul specialistului

**Întrebare:** Am văzut formulare HTML care permit utilizatorului să încarce un fișier în server. Cum pot crea un asemenea formular și cum îl pot folosi cu PHP?

**Răspuns:** Pentru a crea un formular de încărcare, specificați `ENCTYPE="multipart/form-data"` în eticheta `FORM` și includeți un control de introducere a datelor cu atributul `TYPE="FILE"`. Iată un exemplu:

```
<HTML>
<HEAD>
<TITLE>incarca.html</TITLE>
</HEAD>
<BODY>
<H2>Incarcarea fisierelor</H2>
<FORM METHOD="POST" ACTION="incarca.php"
ENCTYPE="multipart/form-data">
Incarca acest fisier:
<INPUT NAME="fisier_utilizator" TYPE="FILE">

<INPUT TYPE="SUBMIT" VALUE="Trimite fisierul">
</FORM>
</BODY>
</HTML>
```

Când utilizatorul apasă pe butonul de trimitere, browserul încarcă fișierul în server. Scriptul dumneavoastră PHP poate obține accesul la calea fișierului încărcat astfel:

```
$HTTP_POST_FILES["fisier_utilizator"]["tmp_name"]
```

unde `fisier_utilizator` este valoarea atributului `NAME` asociat controlului `INPUT` folosit pentru încărcarea fișierului.

La terminarea execuției scriptului, PHP șterge în mod automat fișierul încărcat. Dacă doriți să salvați fișierul, puteți invoca funcția `move_uploaded_file()`. De exemplu:

```
$result =
move_uploaded_file($HTTP_POST_FILES["fisier_utilizator"]
["tmp_name"], "/home/bill/test.txt");
$result = $result ? "true" : "false";
echo "
move_uploaded_file(): $result";
```

Funcția `move_uploaded_file()` vă impune să specificați calea fișierului încărcat și calea unde doriți să fie mutat fișierul. Funcția returnează `true` dacă operația reușește, respectiv `false` în caz contrar.



### Test „la minut”

- Care este valoarea de mod folosită pentru a solicita acces de citire la un fișier?
- Care este funcția folosită pentru închiderea unui fișier?
- Care este funcția folosită pentru a citi o linie de text dintr-un fișier?
- Care este funcția folosită pentru a scrie într-un fișier?

## Utilizarea cataloagelor

În afară de suita sa de funcții destinate lucrului cu fișiere, PHP include și funcții pentru utilizarea cataloagelor. Cele mai importante funcții pentru utilizarea cataloagelor vă permit să obțineți catalogul de lucru și să lucrați cu acesta, să manipulați căi de acces, să citiți conținutul unui catalog, să vizualizați și să modificați privilegiile unui catalog, precum și să creați și să ștergeți cataloage.

### Obținerea și modificarea catalogului de lucru

Funcția `getcwd()` returnează un șir care conține catalogul curent de lucru. Funcția nu necesită argumente și, implicit, poate fi utilizată astfel:

```
$dir = getcwd();
```

Pentru a schimba catalogul curent de lucru, invocați funcția `chdir()`, care are următoarea formă:

Răspunsuri la test:

- "r" sau oricare dintre opțiunile "r+", "w+" sau "a+".
- `fclose()`
- `fgets()`
- `fwrite()`



```
chdir(ume_catalog)
```

unde *ume\_catalog* este calea sau numele catalogului de lucru dorit. Funcția returnează *true* dacă operația reușește; în caz contrar, returnează *false*.

De exemplu, pentru a face din */tmp* catalogul curent de lucru, invocați funcția *chdir()* după cum urmează:

```
$ok = chdir("/tmp");
if (!$ok)
{
 die("Nu a putut schimba catalogul de lucru.");
}
```

## Manipularea căilor de acces

PHP include numeroase funcții utile pentru manipularea căilor de acces. Funcția *dirname()* preia calea de acces la un fișier și returnează toată calea, mai puțin componenta finală a căii specificate. În cazul în care componenta finală este un fișier, funcția vă ajută să izolați numele fișierului de restul căii. De exemplu, dacă se dă calea *"home/bill/bin/script.php"*, funcția va returna *"home/bill/bin"*. Funcția are următoarea formă:

```
dirname(cale)
```

Funcția *basename()* execută operația complementară, returnând numai componenta finală a căii specificate. De exemplu, dacă este dată calea *"home/bill/bin/script.php"*, funcția va returna *"script.php"*. Funcția are următoarea formă:

```
basename(cale)
```

Dacă doriți să executați mai multe operații cu o cale, funcția *pathinfo()* vă poate fi de folos. Această funcție returnează un tablou asociativ care include trei elemente: rezultatul invocării funcției *dirname()* având ca argument calea respectivă, rezultatul invocării funcției *basename()* având ca argument calea respectivă, precum și extensia fișierului (dacă există) asociată rezultatului invocării funcției *basename()* având ca argument calea respectivă.

Iată un exemplu care prezintă modul de utilizare a funcției *pathinfo()*:

```
$info = pathinfo("/home/bill/bin/script.php");
foreach ($info as $nume=>$valoare)
{
 echo "
$nume=>$valoare";
}
```

Datele de ieșire ale exemplului sunt următoarele:

```
dirname=>/home/bill/bin
basename=>script.php
extension=>php
```

## Atenție

Versiunea PHP 4.04pl1 își încheie execuția dacă invocați funcția *pathinfo()* pe o cale din care lipsește extensia fișierului. Se pare că acest defect de program a fost remediat în versiunea PHP 4.05.

## Vizualizarea și modificarea privilegiilor de catalog

Puteți invoca funcțiile rezumate în tabelul 11-1 folosind ca argumente fișiere sau cataloage. Aceste funcții vă permit să vizualizați o varietate de caracteristici ale cataloagelor, inclusiv privilegiile de catalog, care sunt returnate de funcția *fileperms()*.

Similar, puteți invoca funcția *chmod()*, care a fost descrisă anterior, folosind ca argument un fișier sau un catalog. Utilizați această funcție pentru a stabili privilegiile de catalog exact așa cum ați folosi-o pentru a stabili privilegiile de fișier.

## Citirea conținutului unui catalog

PHP furnizează trei funcții care vă permit să citiți conținutul unui catalog, ca și cum catalogul ar fi un fișier. Aceste funcții sunt:

- *opendir()*, care permite citirea unui catalog
- *readdir()*, care citește o intrare dintr-un catalog
- *closedir()*, care închide catalogul, eliberând resursele alocate de funcția *opendir()*

Rezultatul apelării funcției *readdir()* este un șir care conține numele următorului fișier sau sub-catalog al catalogului deschis. Funcția returnează *false* atunci când au fost citite toate intrările din catalog.

Iată un exemplu care prezintă modul de utilizare a acestor funcții pentru citirea conținutului unui catalog:

```
$dh = opendir("/home/bill/www");
if (!$dh)
{
 die("Nu a reusit sa deschida catalogul.");
}
$ss = readdir($dh);
while ($ss)
{
 echo "
$ss";
 $ss = readdir($dh);
}
closedir($dh);
```

Datele de ieșire ale exemplului sunt asemănătoare cu următoarele:

```
..
php
index.html
phpinfo.php
```

## Crearea unui catalog

Pentru a crea un catalog, invocați funcția `mkdir()`, care are următoarea formă:

```
mkdir(nume_catalog, mod)
```

unde *nume\_catalog* este calea sau numele catalogului care urmează a fi creat, iar *mod* atribuie privilegiile care trebuie acordate noului catalog. În mod normal, prima cifră a argumentului *mod* trebuie să fie 0, astfel încât PHP să-l perceapă ca pe o valoare scrisă în octal. Funcția returnează `true` dacă creează catalogul; în caz contrar, returnează `false`. De exemplu, pentru a crea un catalog denumit `test` și pentru a-i atribui privilegiile `rw-r-x-r-x`, invocați funcția `mkdir()` după cum urmează:

```
mkdir("test", 0751)
```

### Sugestie

Puteți folosi funcția `rename()` pentru a modifica numele unui catalog.

## Ștergerea unui catalog

Pentru a șterge un catalog, invocați funcția `rmdir()`, transferându-i ca argument calea sau numele catalogului care urmează a fi șters. De exemplu, pentru a șterge catalogul `/home/bill/vechi`, invocați funcția `rmdir()` după cum urmează:

```
rmdir("/home/bill/vechi")
```

Se pot șterge numai cataloage vide. Funcția va returna zero în cazul producerii unei erori.



### Test „la minut”

- Care este funcția folosită pentru citirea unei întrări dintr-un catalog?
- Care este funcția folosită pentru a obține extensia de fișier asociată unei căi?
- Care este funcția ce returnează calea catalogului curent de lucru?



### Sfatul specialistului

**Întrebare:** Există și alte funcții PHP care operează cu fișiere și cataloage și despre a căror existență ar trebui să știu?

**Răspuns:** Deși acesta este un capitol foarte lung, nu putem să discutăm despre fiecare funcție PHP care lucrează cu fișiere și cataloage. În plus, nu sunt

Răspunsuri la test:

- `readdir()`, `cu opendir()` și `closedir()`
- `pathinfo()`
- `getcwd()`

puține funcțiile cu elemente „ciudate”, care au fost omise din motive de simplitate și concizie. Pentru mai multe informații, examinați manualul PHP pe suport electronic, la adresa [www.php.net](http://www.php.net). Este necesar să procedați astfel mai ales dacă folosiți PHP sub Microsoft Windows. De asemenea, familiarizați-vă cu baza de date cu defecte de program existentă la adresa [www.php.net](http://www.php.net). Acolo veți descoperi cele mai recente informații cu privire la remedii și soluții ocolitoare, care pot influența comportarea funcțiilor care operează cu fișiere și cataloage.



## Proiect 11-3: Un program de răsfoire a agendei cu adrese

În cadrul acestui proiect, veți construi un script care vă permite să parcurgeți un fișier text care conține o agendă de adrese. De asemenea, scriptul vă va permite să adăugați întrări noi în agenda cu adrese.

### Scopurile proiectului

- Prezentarea modului de navigare în fișiere
- Prezentarea modului de utilizare a limbajului PHP pentru a stabili atributul `VALUE` asociat unui control de tip formular

### Pas cu pas

1. Plasați următorul script PHP într-un fișier numit `browser.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Program de navigare în agenda cu adrese</TITLE>
</HEAD>
<BODY>
<H2>Program de navigare în agenda cu adrese</H2>
<FORM METHOD="POST" ACTION="browser.php">
<?php
if (isset($stanga))
{
 $offset -=49;
 if ($offset < 0)
 {
 $offset = 0;
 }

 $fh = fopen("o_carte.txt", "r");
 if (!$fh)
 {
 die("
Nu s-a reusit deschiderea fisierului.");
 }

 fseek($fh, $offset, SEEK_SET);

 $nume = fread($fh, 24);
 $email = fread($fh, 24);
```

```

fclose($fh);
}
elseif(isset($dreapta))
{
 $offset += 49;
 if ($offset < 0)
 {
 $offset = 0;
 }

 $fh = fopen("o_carte.txt", "r");
 if (!$fh)
 {
 die("
Nu s-a reusit deschiderea fisierului.");
 }

 fseek($fh, $offset, SEEK_SET);

 $nume = fread($fh, 24);
 $email = fread($fh, 24);

 fclose($fh);
}
elseif(isset($cauta))
{
 $offset = 0;
 $model = $nume;
 $nume = "";
 $email = "";

 $fh = fopen("o_carte.txt", "r");
 if (!$fh)
 {
 die("
Nu s-a reusit deschiderea fisierului.");
 }

 fseek($fh, $offset, SEEK_SET);

 $s = fgets($fh, 256);
 while(!feof($fh))
 {
 $nume_fisier = fread($fh, 24);
 $email_fisier = fread($fh, 24);
 $identic = strstr($nume_fisier, $model);
 if ($identic !== false) break;
 $offset += 49;
 }

 fclose($fh);

 $nume = $nume_fisier;
 $email = $email_fisier;
}
elseif(isset($adauga))
{
 $nume = str_pad($nume, 24);
 $email = str_pad($email, 24);

```

```

$fh = fopen("o_carte.txt", "a");
if (!$fh)
{
 die("
Nu s-a reusit deschiderea fisierului.");
}
fwrite($fh, $nume);
fwrite($fh, email);
fwrite($fh, "\n");
$offset = ftell($fh) - 49;

fclose($fh);
}
$nume = trim($nume);
$email = trim($email);

?>

Nume:

<INPUT TYPE="TEXT" NAME="nume"
 <?php echo "VALUE=\"\$nume\" " ?>>

Adresa de e-mail:

<INPUT TYPE="TEXT" NAME="e-mail"
 <?php echo "VALUE=\"\$e-mail\" " ?>>

<INPUT TYPE="SUBMIT" NAME="cauta" VALUE="Cauta">
<INPUT TYPE="SUBMIT" NAME="stanga" VALUE="<">
<INPUT TYPE="SUBMIT" NAME="dreapta" VALUE=">">
<INPUT TYPE="SUBMIT" NAME="adauga" VALUE="Adauga">
<INPUT TYPE="HIDDEN" NAME="offset"
 <?php echo "VALUE=\"\$offset\" " ?>>
</FORM>
</BODY>
</HTML>

```

2. Încărcați textul următor în server, atribuindu-i numele o\_carte.txt și plasându-l în același catalog ca și scriptul:

```

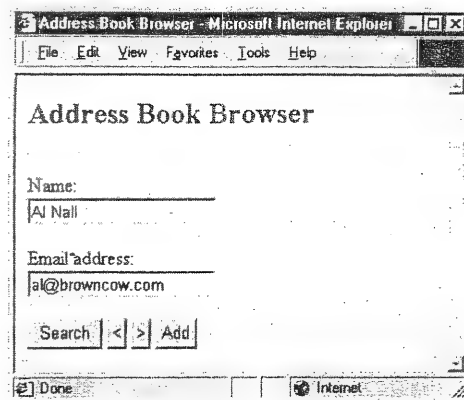
Al Nail al@browncow.com
Bob Tale bob@stories.com
Chuck Stake chuck@beef.com
Ed Nogg ed@beverage.com
Xi Lentz xi@quiet.com
Yo Hoho yo@pirates.com
Zak Cloth zak@ashes.com

```

Fiecare linie a fișierului trebuie să aibă exact 48 de caractere; adresele de e-mail trebuie să înceapă de la coloana 25.

3. Alocăți un timp studiului scriptului PHP. Observați că scriptul conține un formular HTML și că atributul ACTION al etichetei FORM se referă la fișierul script, care combină într-un singur fișier formularul HTML și instrucțiunile PHP care prelucrează datele din formular.
4. Remarcați utilizarea variabilei \$offset pentru a urmări poziția curentă în timpul navigării. Deoarece cele două câmpuri (nume și adresă de e-mail) stocate în fișier au dimensiuni fixe, logica programului este simplă; în cazul în care câmpurile ar fi avut dimensiuni variabile, logica programului ar fi fost sensibil mai complexă.

5. De asemenea, observați modul în care se folosește PHP pentru a stabili atributele VALUE ale casetelor cu text denumite nume și email, pe baza valorilor variabilelor PHP cu același nume.
6. Asigurați-vă că PHP are acces de citire și scriere la fișierul o\_carte.txt. Dacă nu știți sigur cum trebuie să procedați, apăsați la administratorul dumneavoastră de sistem.
7. Orientați un browser Web spre adresa URL a fișierului script. Ecranul browserului trebuie să fie asemănător celui prezentat în continuare. Puteți folosi butoanele pentru a naviga înainte și înapoi în fișier, pentru a căuta un nume care conține un text specificat, respectiv pentru a adăuga o nouă intrare în lista cu adrese.



### Test de evaluare

1. Care este comanda UNIX care șterge catalogul (vid) test?
2. Care sunt privilegiile numerice pe care le veți atribui unui fișier pentru a acorda utilizatorului său numai accesul pentru citire și pentru a nu acorda altor utilizatori nici o categorie de acces?
3. Care este apelul de funcție care deschide fișierul test.txt, acordând accesul de atașare și de citire la un fișier?
4. Care este apelul de funcție care stabilește poziția pointerului fișierului asociat identificatorului \$f la sfârșitul fișierului?
5. Care este apelul de funcție care returnează privilegiile asociate catalogului /test?

## Modulul 12: Expedierea și recepționarea mesajelor de poștă electronică

### Scopuri

- Învățați să expediați mesaje de e-mail prin intermediul protocolului SMTP
- Învățați să obțineți accesul la mesajele de e-mail rezidente pe un server IMAP
- Învățați să manipulați mesaje și dosarele IMAP

În cadrul acestui modul, este explicat modul de utilizare a limbajului PHP pentru a expedia, respectiv pentru a recepționa mesaje de e-mail. Pentru ca funcționalitățile prezentate în acest modul să fie utilizabile, serverul dumneavoastră PHP trebuie configurat astfel încât să accepte IMAP. Mai mult, scripturile dumneavoastră PHP trebuie să fie capabile de a obține accesul la serviciile SMTP (Simple Mail Transfer Protocol) și IMAP (Interim Mail Access Protocol). Așadar, consultați-vă cu administratorul dumneavoastră de sistem înainte de a investi timp în depănarea exemplului; problema o poate constitui serverul PHP, nu dumneavoastră sau exemplul prezentat.

### Expedierea mesajelor de poștă electronică

Configurația PHP standard acceptă expedierea mesajelor de e-mail prin intermediul SMTP (abreviere de la Simple Mail Transfer Protocol). Acesta este protocolul standard folosit pentru transferul mesajelor de e-mail de la un sistem la altul, prin intermediul Internetului.

Mesajele de e-mail sunt alcătuite din două părți: o serie de *antete de mesaj* și un *corp*. Antetele de mesaj indică adresa destinatarului și subiectul mesajului, precum și alte informații. Corpul conține mesajul în sine.

Pentru a expedia un mesaj de e-mail prin intermediul SMTP, invocați funcția `mail()`, care are următoarea formă:

```
mail(destinatar, subiect, corp)
```

unde *destinatar* indică adresa de e-mail a destinatarului, *subiect* specifică antetul de e-mail care conține subiectul mesajului, iar *corp* este corpul mesajului. Se obișnuiește ca adresele de e-mail să includă și adresa de e-mail a expeditorului. Pentru a include adresa de e-mail a expeditorului, folosiți următoarea formă a funcției `mail`:

```
mail(destinatar, subiect, corp, antete)
```

unde *antete* specifică adresele de e-mail suplimentare, precum antetul `from`:

Iată un exemplu care prezintă modul de expediere a unui mesaj de e-mail, care conține un antet ce include adresa de e-mail a expeditorului:

```
mail("bill@osborne.com",
"Acesta este subiectul mesajului",
"Acesta este corpul unui mesaj foarte scurt",
"From: expeditor@osborne.com")
```

Puteți specifica mai mulți destinatari prin separarea fiecărui destinatar de următorul prin intermediul unei virgule:

```
mail("bill@osborne.com", "bob@osborne.com",
"Acesta este subiectul mesajului",
"Acesta este corpul unui mesaj foarte scurt",
"From: expeditor@osborne.com")
```

Funcția mail() returnează true dacă serverul SMTP acceptă mesajul; în caz contrar, returnează false. Rețineți că acceptarea de către serverul SMTP nu garantează transmiterea către destinatar a mesajului dumneavoastră de e-mail. Nu există nici o modalitate 100% sigură de a verifica faptul că mesajul dumneavoastră a fost transmis, așa cum nu există nici o modalitate absolut sigură de a garanta că destinatarul a citit mesajul, l-a înțeles și a fost de acord cu el.

Unele servere SMTP refuză să accepte adrese și antete care conțin spații albe la sfârșit. Dacă scriptul dumneavoastră presupune că utilizatorul va introduce aceste valori, trebuie să invocați funcția trim() având valorile respective ca argument. De exemplu:

```
mail(trim($destinatar), trim($subiect), $corp,
"From: " . trim($expeditor))
```



### Test „la minut”

- Care este funcția PHP care trimite mesaje de poștă electronică prin intermediul serviciului SMTP?
- Când expediați mesaje de e-mail folosind PHP, care este caracterul folosit pentru a separa între ele adresele mai multor destinatari?



### Proiect 12-1: Un script de trimitere a formularelor

În cadrul acestui proiect, veți construi un script care adună date dintr-un formular HTML și le trimite unui utilizator specificat, prin intermediul poștei electronice. Scriptul este scris de așa manieră încât poate fi controlat folosind variabile de formular HTML. Puteți crea o diversitate de formulare HTML care folosesc scriptul, fără a fi necesară revizuirea scriptului.

Răspunsuri la test:

- mail()
- o virgulă



### Sfatul specialistului

**Întrebare:** Este posibilă utilizarea limbajului PHP pentru expedierea mesajelor de poștă electronică în care sunt incluse fișiere atașate?

**Răspuns:** Da, dar biblioteca PHP nu include funcții care facilitează această operație. Dacă fișierul atașat nu este un fișier de tip text simplu, trebuie să-l codificați folosind formatul MIME (Multipurpose Internet Mail Extensions). Apoi, puteți încorpora în antete speciale datele codificate în format MIME. Diferiți programatori și autori au scris funcții și clase PHP destinate a simplifica acest proces. Pentru a învăța mai multe despre fișiere atașate și MIME, vizitați adresa <http://www.phpbuilder.com>.

### Scopurile proiectului

- Prezentarea modului de expediere a mesajelor de poștă electronică folosind PHP
- Prezentarea modului de creare a unui script reutilizabil care prelucrează datele din formularul HTML

### Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit mailform.php și încărcați acest fișier în serverul dumneavoastră PHP:

```
<?php
$corp = "";
foreach ($HTTP_POST_VARS as $nume=>$valoare)
{
 $corp = $corp . "$nume=>$valoare\n";
}
mail($mailform_destinatar, $mailform_subiect, $corp,
"From: " . $mailform_expeditor);
header("Location: $mailform_adresa");

?>
```

2. Alocați un timp studiului scriptului PHP. Scriptul obține acces la valorile câmpurilor din formular prin intermediul tabloului asociativ \$HTTP\_POST\_VARS. Apoi, scriptul comasează valorile tuturor câmpurilor în variabila șir \$corp, care este folosită pentru a stabili valoarea corpului mesajului de e-mail.

De asemenea, remarcați și celelalte variabile din formular, și anume \$mailform\_destinatar, \$mailform\_expeditor și \$mailform\_subiect. Stabilind valorile controalelor asociate ale formularului, puteți specifica adresa de e-mail a expeditorului și a destinatarului, precum și linia de subiect a mesajului de e-mail.

Observați că scriptul folosește funcția PHP header(), care trimite browserului un antet HTTP. Antetul Location (locăție) determină browserul să preia și să afișeze documentul asociat cu adresa URL specificată. Adresa URL în sine este specificată de un câmp ascuns al formularului HTML care trimite date scriptului. Prin stabilirea valorii acestui câmp, puteți specifica pagina pe care o vede utilizatorul după ce datele din formular au fost prelucrate. În mod caracteristic, veți trimite utilizatorul la o pagină în care acestuia i se mulțumește pentru datele introduse sau pentru tranzacție.

3. Plasați următoarea pagină HTML într-un fișier denumit mailfrm1.html și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul mailform.php:

```
<HTML>
<HEAD>
<TITLE>mailfrm1.html</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="mailform.php">

<H2>Formular de trimitere a datelor pentru e-mail</H2>
Introduceti aici datele dumneavoastra:

<INPUT TYPE="TEXT" NAME="data" SIZE="24">

<INPUT TYPE="SUBMIT" VALUE="Trimite">

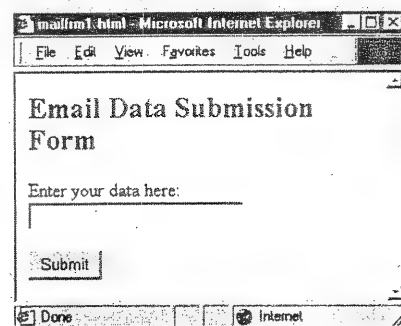
<INPUT TYPE="HIDDEN" NAME="mailform_destinatar"
VALUE="bmccarty@apu.edu">
<INPUT TYPE="HIDDEN" NAME="mailform_expeditor"
VALUE="bmccarty@apu.edu">
<INPUT TYPE="HIDDEN" NAME="mailform_subiect"
VALUE="Subiectul">
<INPUT TYPE="HIDDEN" NAME="mailform_adresa"
VALUE="mailfrm2.html">

</FORM>
</BODY>
</HTML> HYPERLINK
```

4. Observați că formularul HTML are un singur câmp pentru introducerea datelor de către utilizator, denumit data. Cu toate acestea, scriptul va prelucra toate câmpurile pe care doriți să le definiți.
5. Plasați următoarea pagină HTML într-un fișier numit mailfrm2.html și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul mailform.php:

```
<HTML>
<HEAD>
<TITLE>mailfrm2.html</TITLE>
</HEAD>
<BODY>
<H2>Mesaj receptionat</H2>
Multumim! Datele dumneavoastra au fost trimise!
</BODY>
</HTML>
```

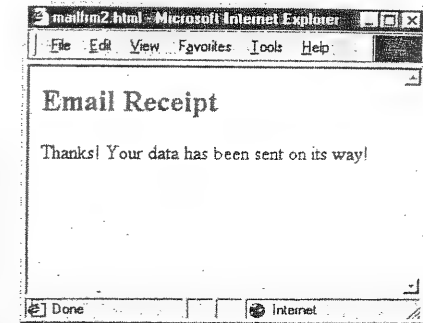
6. Orientați un browser Web către adresa URL a scriptului mailfrm1.html. Browserul afișează următorul ecran:



7. Introduceți datele aferente formularului și executați clic pe butonul de trimitere. Browserul afișează ecranul alăturat:
8. Mesajul primit de destinatar este similar cu următorul listing:

```
Data: Sun, 15 Apr 2005 16:27:11 -0700
Expeditor: bill@osborne.com
Destinatar: bill@osborne.com
Subiect: Subiectul
```

```
data=Acestea sunt datele pe care le-am
introdus.
mailform_destinatar=bill@osborne.com
mailform_expeditor=bill@osborne.com
mailform_subiect=Subiectul
mailform_adresa=mailfrm2.html
```



## Recepționarea mesajelor de e-mail

Această secțiune tratează elementele fundamentale ale procesului de recepționare prin intermediul IMAP, protocolul Interim Mail Access Protocol. Din păcate, recepționarea mesajelor de e-mail este un proces oarecum mai complex decât expedierea lor; ca atare, această secțiune este semnificativ mai lungă decât precedenta.

Mai mult, deși SMTP face parte din configurația PHP standard, IMAP nu este o componentă a acestei configurații. Deci, dacă administratorul dumneavoastră PHP nu a configurat PHP astfel încât să lucreze cu IMAP, exemplele din această secțiune nu vor funcționa.

## Deschiderea unei cutii poștale

Un server IMAP este accesibil în același mod ca și un fișier oarecare. Mai întâi, trebuie să deschideți o conexiune cu serverul. Apoi, puteți trimite cereri serverului și puteți primi răspunsuri de la acesta. Când ați terminat de utilizat serverul, trebuie să închideți conexiunea.

Pentru a deschide o conexiune cu un server IMAP, folosiți funcția `imap_open()`, care are următoarea formă:

```
imap_open(cutie_postala, identificador_utilizator, parola)
```

Argumentul numit *cutie\_postala* specifică patru elemente informaționale:

- Numele gazdei sau adresa IP a serverului IMAP
- Protocolul care va fi utilizat (IMAP)
- Portul care se va folosi pentru contactarea serverului (în general 143)
- Cutia poștală care va fi deschisă (în general INBOX)



Argumentul folosește numeroși delimitatori pentru a separa un element de altul. Iată o valoare caracteristică a argumentului:

```
{localhost/imap:143}INBOX
```

În cadrul acestui exemplu, numele gazdei este specificat sub forma localhost; acest lucru este posibil când un server PHP și un server IMAP rulează pe același sistem gazdă. În caz contrar, trebuie să specificați numele gazdei care rulează serverul IMAP. De exemplu:

```
{mail.osborne.com/imap:143}INBOX
```

Celelalte argumente ale funcției `imap_open()`, *identificator\_utilizator* și *parola*, specifică identificatorul de utilizator și parola folosite pentru a obține accesul la serverul IMAP. Așa cum funcția `fopen()` returnează un identificator pe care îl puteți folosi pentru a obține accesul la un fișier, funcția `imap_open()` returnează un identificator pe care îl puteți folosi pentru a obține accesul la serverul IMAP. Dacă PHP nu poate deschide o conexiune cu serverul IMAP, funcția `imap_open()` returnează false.

### Sugestie

Dacă un script se încheie cu mesajul „Call to undefined function: imap\_open”, aceasta arată că PHP nu a fost configurat pentru a folosi IMAP. Luați legătura cu administratorul dumneavoastră de sistem, pentru a beneficia de asistență în rezolvarea acestei probleme.

Iată un exemplu care prezintă modul de invocare a funcției `imap_open()` și testarea rezultatului returnat de funcție:

```
function open_mailbox**($server,
 $identificator_utilizator, $parola)
{
 echo "\n\n";
 echo "<H4>Deschide conexiunea IMAP cu $server.</H4>";
 $identificator = imap_open($server . "INBOX",
 $identificator_utilizator, $parola);
 if ($identificator === false)
 {
 echo "Nu s-a putut deschide cutia postala IMAP INBOX.";
 print_error_stack();
 die();
 }
 return $identificator;
}
```

Exemplul este prezentat sub forma unei funcții definite de utilizator, denumită `open_mailbox()`. Puteți folosi această funcție în propriile dumneavoastră scripturi PHP. Funcția `open_mailbox()` generează date de ieșire care vă vor ajuta să fiți la

curent cu acțiunile programului dumneavoastră. De asemenea, se execută o verificare a apariției erorilor. Veți economisi un oarecare volum de muncă dacă apelați funcția `open_mailbox()` în loc să apelați direct funcția `imap_open()`. Desigur, dacă datele generate de funcția `open_mailbox()` sau modalitatea de verificare a apariției erorilor nu vă satisfac, le puteți modifica, vă puteți scrie propria dumneavoastră funcție definită de utilizator sau apelați direct funcția `imap_open()`.

### Sugestie

Dacă nu vă mai amintiți foarte exact de funcțiile PHP definite de utilizator, trebuie să treceți din nou în revistă Modulul 7.

Observați că funcția `open_mailbox()` apelează funcția `print_error_stack()` dacă apelul la funcția `imap_open()` eșuează. Funcția `print_error_stack()` este o altă funcție definită de utilizator. Iată care este definiția acesteia:

```
function print_error_stack()
{
 echo "\n\n";
 echo "<H4>Stiva de erori IMAP</H4>";
 $erori = imap_errors();
 if ($erori)
 {
 foreach ($erori as $cheie=>$valoare)
 {
 echo "\n$cheie: $valoare";
 }
 }
}
```

Funcția apelează funcția de bibliotecă IMAP `imap_errors()`, care returnează un tablou care conține mesaje de eroare. Apoi, funcția parcurge iterativ tabloul, folosind o instrucțiune `foreach`, care afișează fiecare eroare.

### Comutarea între dosare

În afară de dosarul INBOX standard, IMAP permite unui utilizator să definească dosare, care pot fi folosite de acesta din urmă pentru stocarea și organizarea mesajelor primite. În orice moment, un dosar IMAP – INBOX sau un alt dosar – este considerat ca fiind dosarul curent. După ce ați stabilit o conexiune cu un server IMAP, puteți cere serverului să desemneze alt dosar ca dosar curent. Iată o funcție definită de utilizator care vă permite să procedați astfel:

```
function switch_to_folder**($identificator, $server, $prefix, $dosar)
{
 if ($dosar == "INBOX")
```

\* În traducere *apel la o funcție nedefinită* – N.T.

\*\* În traducere *deschide cutia poștală* – N.T.

\* În traducere *afișează stiva cu erori* – N.T.

\*\* În traducere *comută la dosar* – N.T.

```

{
 $dosar = $server . "INBOX";
}
else
{
 $dosar = $server . $prefix . "INBOX." . $dosar;
}
echo "\n\n";
echo "<H4>Comuta în dosarul $dosar</H4>";
$ok = imap_reopen($identificator, $dosar);
if ($ok === false)
{
 echo "\nNu a putut deschide dosarul specificat.";
 print_error_stack();
}
}

```

Funcția definită de utilizator invocă funcția IMAP `imap_reopen()`, care preia două argumente: identificatorul returnat de funcția `imap_open()` (sau de către funcția definită de utilizator `open_mailbox()`, care apelează funcția `imap_open()`) și numele dosarului care urmează a fi deschis. Funcția definită de utilizator preia, în afară de identificator, numeroase alte argumente, pe care le folosește pentru a alcătui numele dosarului. Puteți apela funcția `switch_to_folder()` astfel:

```
switch_to_folder("{localhost/imap:143}", "~/mail/", "dosarul")
```

## Notă

De obicei, numele dosarelor IMAP sunt obținute prin prefixarea numelui dosarului cu particula `~/mail/` și `INBOX`, așa cum o presupune funcția `switch_to_folder()` și invocarea dată ca exemplu. Cu toate acestea, un administrator de sistem poate configura o altă politică de atribuire a numelor. Dacă aveți o problemă cu utilizarea funcției `switch_to_folder()`, luați legătura cu administratorul dumneavoastră de sistem, pentru a determina politica adecvată de denumire a dosarelor.

## Închiderea unei cutii poștale

Când ați terminat de utilizat un server IMAP, trebuie să îl închideți, așa cum închideți un fișier atunci când ați finalizat utilizarea acestuia. Iată o funcție definită de utilizator pentru închiderea unei conexiuni IMAP:

```

function close_mailbox($identificator)
{
 echo "\n\n";
 echo "<H4>Inchide conexiunea IMAP</H4>";
 $ok = imap_close($identificator);
 if ($ok === false)
 {
 echo "\nNu a reusit sa inchida cutia postala.";
 print_error_stack();
 }
}

```

Funcția folosește funcția din biblioteca IMAP `imap_close()` și funcția definită de utilizator `print_error_stack()` care a fost prezentată anterior.

## Obținerea informațiilor referitoare la o cutie poștală

După deschiderea unei conexiuni IMAP, puteți obține acces la informații care descriu cutia poștală curentă. De exemplu, iată o funcție definită de utilizator care returnează numărul mesajelor din cutia poștală curentă:

```

function get_message_count*($identificator)
{
 return imap_num_msg($identificator);
}

```

Funcția definită de utilizator nu face decât să apeleze funcția din biblioteca IMAP `imap_num_msg()`. Totuși, puteți adăuga linii de program suplimentare la funcția definită de utilizator; de exemplu, puteți adăuga o instrucțiune care afișează numărul de mesaje.

Iată o funcție definită de utilizator mai complexă, care afișează o varietate de informații referitoare la cutia poștală curentă:

```

function print_mailbox_status*($identificator)
{
 echo "\n\n";
 echo "<H4>Starea cutiei postale</H4>";
 $n = imap_num_msg($identificator);
 echo "\nCutia postala contine $n mesaje.";

 $n = imap_num_recent($identificator);
 echo "\nCutia postala contine $n mesaje recente.";

 echo "\n\n";
 $obiect_cutie_postala = imap_mailboxmsginfo($identificator);
 if ($obiect_cutie_postala)
 {
 $tablou_cutie_postala = get_object_vars($obiect_cutie_postala);
 foreach ($obiect_cutie_postala as $cheie=>$valoare)
 {
 echo "\n$cheie: $valoare";
 }
 }
}

```

Această funcție afișează numărul mesajelor și pe acela al mesajelor recente din cutia poștală. Apoi, funcția afișează o varietate de informații, inclusiv:

- Data ultimei modificări a cutiei poștale
- Numele cutiei poștale

\* În traducere *obține numărul de mesaje* – N.T.

\*\* În traducere *afișează starea cutiei poștale* – N.T.

- Numărul mesajelor din cutia poștală
- Numărul mesajelor recente din cutia poștală
- Numărul mesajelor necitite din cutia poștală
- Numărul mesajelor șterse din cutia poștală
- Dimensiunea cutiei poștale, în octeți

Observați funcția de bibliotecă IMAP `imap_mailboxmsginfo()`, care returnează o valoare atribuită obiectului `$obiect_cutie_postala`. Această valoare este de tip obiect. Modulul 15 va aborda obiectele și modul de utilizare a acestora. Totuși, nu trebuie să cunoașteți modul de lucru cu obiectele pentru a folosi această valoare. Prin transferul valorii ca argument al funcției `get_object_vars()`, aceasta returnează un tablou asociativ care conține datele obiectului. Funcția definită de utilizator `print_mailbox_status()` folosește această tehnică pentru a obține și a afișa informații referitoare la starea cutiei poștale.

Iată un rezultat caracteristic al funcției `print_mailbox_status()`:

```
Starea cutiei postale
INBOX contine 3 mesaje
INBOX contine 0 mesaje recente.
```

```
Unread: 0
Deleted: 0
Nmsgs: 3
Size: 1078
Date: Fri, 25 May 2001 08:57:53 -0700 (PDT)
Driver: imap
Mailbox: {localhost.localdomain:143/imap/user="bmccarty"}INBOX
Recent: 0
```

Funcția definită de utilizator `dump_mailbox_status()` prezintă o altă metodă de abordare a funcției `imap_mailboxmsginfo()`:

```
function dump_mailbox_status($identificator)
{
 echo "\n\n";
 echo "<H4>Afișarea stării cutiei postale</H4>";
 echo "\n\n";
 $obiect_cutie_postala = imap_mailboxmsginfo($identificator);
 print_r($obiect_cutie_postala);
}
```

Funcția PHP `print_r()` afișează valoarea unui obiect. Această metodă este cu mult mai simplă decât cea folosită de funcția `print_mailbox_status()`, dar formatul datelor de ieșire este cu mult mai puțin inteligibil:

```
Afișarea stării cutiei postale
stdClass Object
(
 [Unread] => 0
 [Deleted] => 0
 [Nmsgs] => 3
```

```
[Size] => 1078
[Date] => Fri, 25 May 2001 08:57:53 -0700 (PDT)
[Driver] => imap
[Mailbox] =>
{localhost.localdomain:143/imap/user="bmccarty"}INBOX
[Recent] => 0
)
```

## Sugestie

Motivul pentru care prezentăm funcțiile IMAP în asociere cu funcțiile definite de utilizator este acela că funcțiile definite de utilizator servesc drept instrumente de construcție a unui program IMAP. Funcțiile definite de utilizator generează date de ieșire și mesaje de eroare care vă ajută să stabiliți o logică adecvată a programului dumneavoastră. După ce, în esență, programul a ajuns să realizeze ceea ce doriți dumneavoastră, puteți ajusta funcțiile definite de utilizator pentru a elimina datele de ieșire, pentru a le re-formata sau orice altceva. O modificare pe care trebuie să o aveți în vedere constă în prefixarea apelurilor de funcții cu un caracter `@`, pentru ca mesajele de eroare prestabilite să nu vă „strice” paginile în cazul apariției unor probleme.

## Obținerea unei liste de mesaje

Iată o funcție definită de utilizator care afișează o listă de mesaje în dosarul curent:

```
function list_messages*($identificator)
{
 echo "\n\n";
 echo "<H4>Antete de mesaje in cutia postala curenta</H4>";
 $antete = imap_headers($identificator);

 if ($antete == false)
 {
 echo "\nNu a reusit sa afiseze mesajele.";
 print_error_stack();
 }
 else
 {
 foreach ($antete as $scheie=>$valoare)
 {
 echo "\n$valoare";
 }
 }
}
```

Funcția definită de utilizator apelează funcția IMAP `imap_headers()`, care returnează un tablou unde fiecare element descrie un mesaj din dosarul curent.

\* În traducere afișează mesaje – N.T.

## Lucrul cu identificatori de mesaje

O ciudățenie a serviciului IMAP constă în aceea că fiecare mesaj dintr-un dosar are atât un număr, cât și un identificator. Numărul reprezintă poziția mesajului în dosarul respectiv; această valoare se poate modifica la adăugarea mesajelor, respectiv la ștergerea mesajelor din dosar. Pe de altă parte, identificatorul unui mesaj nu se modifică niciodată.

IMAP furnizează funcții care vă permit să determinați identificatorul unui mesaj dacă este dat numărul său și invers. Iată funcțiile definite de utilizator care invocă funcțiile de bibliotecă IMAP conexe:

```
function get_message_id*($identificator, $numar)
{
 return imap_uid($identificator, numar);
}
function get_message_num**($identificator, $id)
{
 return imap_msgno($identificator, $id);
}
```

## Obținerea de informații referitoare la un mesaj

O limitare a funcției `imap_headers()`, folosită în funcția definită de utilizator `list_messages()`, este aceea că nu separă fiecare caracteristică a mesajului în câmpuri distincte, astfel încât acestea să fie ușor accesibile pentru un script. Funcția de bibliotecă IMAP `imap_fetch_overview()` returnează un tablou asociativ care descrie un mesaj. Iată o funcție definită de utilizator, care folosește funcția `imap_fetch_overview()` pentru a afișa informațiile despre mesaje:

```
function print_overview***($identificator, $numar)
{
 echo "\n\n";
 echo "<H4>Mesaj $n: Vedere de ansamblu</H4>";
 $nr_mesaj = "$numar";
 $mesaje = imap_fetch_overview($identificator, $nr_mesaj, 0);
 foreach ($mesaje as $mesaj)
 {
 $n = $mesaj->msgno;
 foreach ($mesaj as $proprietate=>$valoare)
 {
 echo "\n$proprietate: $valoare;";
 }
 }
}
```

\* În traducere *obține identificatorul mesajului* – N.T.

\*\* În traducere *obține numărul mesajului* – N.T.

\*\*\* În traducere *afișează vederea de ansamblu* – N.T.

Al doilea argument al funcției `imap_fetch_overview()` vă permite să specificați o listă sau un domeniu de mesaje pentru care funcția returnează vederi de ansamblu. Funcția definită de utilizator `print_overview()` alcătuiește o listă cu un singur membru și transmite lista funcției `imap_fetch_overview()`.

Datele de ieșire conțin următoarele informații:

- Subiectul mesajului
- Numele și adresa de e-mail ale expeditorului
- Data la care a fost trimis mesajul
- Numărul mesajului
- Identificatorul mesajului
- Dimensiunea mesajului (în octeți)
- Indicatoare care precizează dacă:
  - Mesajul este recent
  - Mesajul a fost citit
  - Mesajul a primit un răspuns
  - Mesajul a fost marcat în vederea ștergerii
  - Mesajul este o ciornă

## Obținerea corpului unui mesaj

Corpul unui mesaj include conținutul efectiv al mesajului. Iată o pereche de funcții definite de utilizator care afișează corpul mesajului:

```
function print_body_by_num*($identificator, $numar)
{
 $id = get_message_id($identificator, $numar);
 print_body_id($identificator, $id);
}
function print_body_by_id**($identificator, $numar)
{
 echo "\n\n";
 $numar = get_message_num($identificator, $id);
 $echo "<H4>Message ID $id (Number $numar) Body</H4>";
 $s = imap_body($identificator, $id, FT_UID);
 echo "\n$s";
}
```

Una dintre funcții vă permite să specificați mesajul în funcție de numărul mesajului, iar cealaltă vă permite să specificați mesajul în funcție de identificatorul acestuia. Remarcați modul de implementare a funcției `print_body_by_num()`. Aceasta convertește numărul mesajului într-un identificator de mesaj și apoi apelează funcția `imap_fetch_overview()`, în speță `print_body_by_id()`, care la rândul său apelează funcția IMAP `imap_body()`.

\* În traducere *afișează corpul în funcție de număr* – N.T.

\*\* În traducere *afișează corpul în funcție de identificator* – N.T.

## Obținerea antetelor de mesaj

Antetele de mesaj conțin informații importante, care în general nu apar în corpul mesajului, precum data și subiectul mesajului. Iată o funcție definită de utilizator care afișează antetele unui mesaj, dacă este dat numărul mesajului:

```
function print_headers*($identificator, $numar)
{
 echo "\n\n";
 echo "<H4>Mesaj $n: Antete</H4>";
 $obiect_antet = imap_headerinfo($identificator, $numar);
 $antete = get_object_vars($obiect_antet);
 foreach ($antete as $proprietate=>$valoare)
 {
 if (!is_array($valoare))
 {
 echo "\n$proprietate: $valoare";
 }
 else
 {
 foreach ($valoare as $sub_valoare)
 {
 echo "\n$proprietate: ";
 $sub_valori = get_object_vars($sub_valoare);
 foreach ($sub_valori as $articol=>$valoare_articol)
 {
 echo "\n $articol=>$valoare_articol";
 }
 }
 }
 }
}
```

Similar funcției definite de utilizator `print_mailbox_status()`, funcția `print_headers()` folosește funcția `get_object_vars()` pentru a converti un obiect într-un tablou asociativ. Obiectul returnat de funcția de bibliotecă IMAP `imap_headerinfo()` include următoarele informații:

- Data expedierii mesajului
- Subiectul mesajului
- Identificatorul mesajului la care s-a răspuns prin acest mesaj, dacă există
- Indicatoare de mesaj, cum sunt cele returnate de funcția `imap_fetch_overview()`
- Numele și adresa de e-mail ale:
  - Expeditorului
  - Persoanei care primește răspunsul, dacă este specificată
  - Destinatariilor
  - Destinatariilor eventualelor copii (cc:)
  - Destinatariilor eventualelor copii la indigo necunoscute (bcc:), dacă informațiile respective sunt disponibile

\* În traducere *afișează antetele* – N.T.

Numeroase elemente ale tabloului asociativ au valori de tip tablou. Pentru parcurgerea iterativă și afișarea valorii acestor elemente se folosește o buclă imbricată.

Pentru aplicații mai puțin pretențioase, funcția `dump_headers()` formatează aceleași date de ieșire ca și `print_headers()`, dar folosește pentru aceasta funcția `print_r()`, generând astfel date de ieșire mai puțin inteligibile:

```
function dump_headers($identificator, $numar)
{
 $antete = imap_headerinfo($identificator, $numar);
 echo "\n\n";
 echo "<H4>Mesaj $n: Afișare antete</H4>";
 print_r($antete);
}
```

## Marcarea unui mesaj în vederea ștergerii

Pentru a șterge un mesaj IMAP, mai întâi îl marcați în vederea ștergerii și apoi îl eliminați. Mesajele marcate pentru ștergere, dar care nu au fost încă eliminate, sunt numai semnalate ca șterse și sunt în continuare accesibile.

Iată diferite funcții definite de utilizator care vă permit să ștergeți un mesaj prin specificarea numărului sau a identificatorului mesajului:

```
$function delete_message_by_num*($identificator, $numar)
{
 $id = get_message_id($identificator, $numar);
 delete_message_by_id**($identificator, $id);
}

function delete_message_by_id($identificator, $id)
{
 echo "\n\n";
 $numar = get_message_num($identificator, $id);
 echo "<H4>Marking Message ID $id (Numar $numar) for deletion</H4>";
 $ok = imap_delete($identificator, $id, FT_UID);
 if ($ok == false)
 {
 echo "\nNu s-a reusit stergerea mesajului.";
 print_error_stack();
 }
}
```

Pentru a elimina mesajele șterse, invocați următoarea funcție definită de utilizator:

```
function expunge_messages***($identificator)
{
 echo "\n\n";
 echo "<H4>Elimina mesajele sterse</H4>";
 $ok = imap_expunge($identificator);
 if ($ok == false)
```

\* În traducere *șterge mesajul în funcție de număr* – N.T.

\*\* În traducere *șterge mesajul în funcție de identificator* – N.T.

\*\*\* În traducere *elimină mesaje* – N.T.

```

{
 echo "\nNu s-a reusit eliminarea mesajelor.";
 print_error_stack();
}

```

## ? Sfatul specialistului

**Întrebare:** Furnizorul meu de servicii Internet dispune de un server POP, nu de un server IMAP. Pot avea acces la cutia mea poștală POP cu ajutorul sistemului PHP?

**Răspuns:** Da. De fapt, puteți avea acces la cutia poștală folosind biblioteca IMAP. Pur și simplu deschideți o conexiune cu serverul POP specificând o cutie poștală POP, astfel:

```
$mbox = imap_open("{localhost/pop3:110}INBOX", $identificator_utilizator, $parola);
```

Numele gazdei, protocolul și numărul portului sunt asemănătoare cu valorile similare folosite pentru conectarea la un server IMAP. Ca și în cazul unui server IMAP, poate că este necesară modificarea numelui cutiei poștale, pentru ca acesta să se conformeze politicilor stabilite de către administratorul de sistem.

De asemenea, puteți folosi biblioteca IMAP pentru a vă conecta la un server de informații folosind NNTP (Network News Transfer Protocol). Pentru aceasta, deschideți o conexiune astfel:

```
$nnntp = imap_open("{localhost/nnntp:119}comp.test", "", "");
```

Consultați manualul PHP pe suport electronic la adresa <http://www.php.net> pentru mai multe informații despre utilizarea funcțiilor din biblioteca IMAP pentru a obține accesul la un server POP sau la un server de informații.



## Test „la minut”

- Ce trebuie să faceți înainte de a obține accesul la o cutie poștală IMAP?
- Când sunt marcate pentru ștergere, mesajele IMAP dispar sau nu?
- Care din cei doi identificatori este mai durabil: un număr de mesaj IMAP sau un identificator de mesaj?
- Care sunt informațiile incluse într-un antet de mesaj e-mail?

Răspunsuri la test:

- Deschideți o conexiune cu serverul IMAP.
- Nu. Mesajele IMAP rămân în dosar până când sunt eliminate.
- Identificatorul de mesaj este mai durabil.
- Data la care a fost trimis mesajul, subiectul mesajului, identificatorul mesajului la care s-a răspuns prin mesajul curent (dacă există), indicatoare (flag) de mesaj, respectiv numele și adresele de e-mail ale persoanelor vizate de mesajul e-mail, și anume expeditorul și destinatarul.



## Proiect 12-2: Un program de navigare pentru poșta electronică

În cadrul acestui proiect, veți construi un script care vă permite să parcurgeți un dosar IMAP.

### Scopurile proiectului

- Prezentarea modului de invocare a funcțiilor IMAP prin intermediul funcțiilor definite de utilizator amintite în acest modul
- Prezentarea modului de acces la dosarele și mesajele IMAP

### Pas cu pas

1. Plasați următorul script PHP într-un fișier denumit `cititor.php` și încărcați acest fișier în serverul dumneavoastră PHP:

```

<?php
require("cititor.inc");

echo "<PRE>";

$server = "{ " . $gazda . "/imap:143}";
$imap = open_mailbox($server, $id_utilizator, $parola);
list_messages($imap);
$n = get_message_count($imap);
for ($i = 1; $i <= $n; $i++)
{
 print_headers($imap, $i);
 print_body_by_num($imap, $i);
}
close_mailbox($imap);

echo "</PRE>";

```

2. Plasați următorul script PHP (cam lung) într-un fișier denumit `cititor.inc` și încărcați acest fișier în serverul dumneavoastră, inserându-l în același catalog ca și fișierul `cititor.php`:

```

function open_mailbox ($server, $identificator_utilizator, $parola)
{
 echo "\n\n";
 echo "<H4>Deschide conexiunea IMAP cu $server.</H4>";
 $identificator = imap_open($server, "INBOX", $identificator_utilizator, $parola);
 if ($identificator === false)
 {
 echo "Nu s-a putut deschide cutia postala IMAP INBOX.";
 print_error_stack();
 die();
 }
 return $identificator;
}

function print_error_stack()
{
 echo "\n\n";
 echo "<H4>Stiva de erori IMAP</H4>";
}

```



```

$erori = imap_errors();
if ($erori)
{
 foreach ($erori as $scheie=>$valoare)
 {
 echo "\n$scheie: $valoare";
 }
}

function list_messages ($identificator)
{
 echo "\n\n";
 echo "<H4>Antete de mesaje in cutia postala curenta</H4>";
 $antete = imap_headers($identificator);

 if ($antete == false)
 {
 echo "\nNu a reusit sa afiseze mesajele.";
 print_error_stack();
 }
 else
 {
 foreach ($antete as $scheie=>$valoare)
 {
 echo "\n$valoare";
 }
 }
}

function get_message_count($identificator)
{
 return imap_num_msg($identificator);
}

function print_headers ($identificator, $numar)
{
 echo "\n\n";
 echo "<H4>Mesaj $n: Antete</H4>";
 $obiect_antet = imap_headerinfo($identificator, $numar);
 $antete = get_object_vars($obiect_antet);
 foreach ($antete as $proprietate=>$valoare)
 {
 if (is_array($valoare))
 {
 echo "\n$proprietate: $valoare";
 }
 else
 {
 foreach ($valoare as $sub_valoare)
 {
 echo "\n$proprietate: ";
 $sub_valori = get_object_vars($sub_valoare);
 foreach ($sub_valoare as $articol=>$valoare_articol)
 {
 echo "\n $articol=>$valoare_articol";
 }
 }
 }
 }
}

```

```

 }
}

function print_body_by_num ($identificator, $numar)
{
 $sid = get_message_id($identificator, $numar);
 print_body_id($identificator, $sid);
}

function print_body_by_id ($identificator, $numar)
{
 echo "\n\n";
 $numar = get_message_num($identificator, $sid);
 $echo "<H4>Message ID (Number $numar) Body</H4>";
 $s = imap_body($identificator, $sid, FT_UID);
 echo "\n$s";
}

function get_message_id ($identificator, $numar)
{
 return imap_uid($identificator, $numar);
}

function get_message_num ($identificator, $sid)
{
 return imap_msgno($identificator, $sid);
}

function close_mailbox($identificator)
{
 echo "\n\n";
 echo "<H4>Inchide conexiunea IMAP</H4>";
 $ok = imap_close($identificator);
 if ($ok === false)
 {
 echo "\nNu a reusit sa inchida cutia postala.";
 print_error_stack();
 }
}

```

3. Plasați următoarea pagină HTML într-un fișier numit cititor.html și încărcați acest fișier în serverul dumneavoastră PHP, inserându-l în același catalog ca și scriptul cititor.php și cititor.inc:

```

<HTML>
<HEAD>
<TITLE>cititor.html</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="cititor.php">

<H2>Formular de acces la cutia postală IMAP</H2>
Identificator de utilizator:

<INPUT TYPE="TEXT" NAME="idutilizator" SIZE="24">

```

```


Parola:

<INPUT TYPE="PASSWORD" NAME="parola" SIZE="24">

Server:

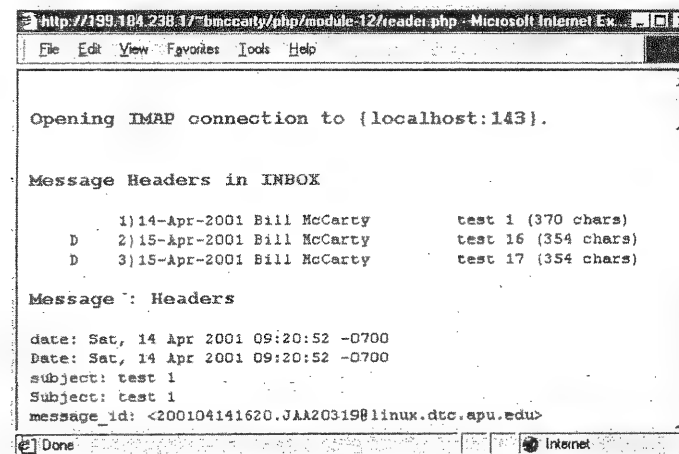
<INPUT TYPE="TEXT" NAME="gazda" SIZE="24" VALUE="localhost">

<INPUT TYPE="SUBMIT" VALUE="Trimite">

</FORM>
</BODY>
</HTML>
```

- Alocați un timp studiului paginii HTML cititor.html. Observați că utilizatorului îi este permis să specifice informațiile minime necesare pentru a obține accesul la o cutie poștală IMAP: numele gazdei serverului IMAP, precum și identificadorul de utilizator și parola utilizatorului.
- Alocați un timp studiului scriptului PHP cititor.php. Remarcați simplitatea scriptului. Acesta:
  - Deschide o conexiune cu serverul IMAP
  - Afișează mesajele din dosarul prestabilit
  - Obține numărul mesajelor din dosarul prestabilit
  - Afișează antetul și corpul fiecărui mesaj
  - Închide conexiunea cu serverul IMAP
 Observați că scriptul folosește o instrucțiune require pentru a încorpora conținutul fișierului cititor.inc.
- Alocați un timp studiului scriptului PHP cititor.inc. Observați că acest script este alcătuit numai dintr-o colecție de funcții definite de utilizator, prezentate anterior în cadrul capitolului de față. Nici una dintre funcții nu a suferit modificări. Dacă știți care este utilitatea funcțiilor respective, atunci acest script nu trebuie să vă preocupe prea mult; puteți înțelege aplicația studiind scriptul cititor.php.
 

Ca ajutor pentru redactarea propriilor dumneavoastră scripturi de e-mail, situl Web aferent acestei cărți include fișierul imap.inc, care conține toate funcțiile definite de utilizator date în acest modul. Pur și simplu inserați o copie a acestui fișier în catalogul dumneavoastră de scripturi, scrieți o instrucțiune require prin care încorporați acest script în propriul dumneavoastră script, după care apelați funcțiile definite de utilizator pe care le definește scriptul.
- Orientați un browser Web spre adresa URL a paginii Web cititor.html. Browserul va afișa un ecran similar cu ilustrația alăturată:
- Introduceți informațiile adecvate pentru serverul dumneavoastră IMAP și executați clic pe butonul „Trimite”. Scriptul trebuie să aibă acces la cutia dumneavoastră poștală IMAP și trebuie să-i afișeze caracterele. Rezultatul trebuie să fie asemănător cu datele prezentate în ilustrația următoare:



## Manipularea dosarelor

Dincolo de accesul la dosare și mesaje, puteți manipula dosare prin utilizarea bibliotecii IMAP. În această secțiune veți afla cum trebuie să procedați.

### Afișarea dosarelor existente

Iată o funcție definită de utilizator care afișează dosarele IMAP disponibile:

```
function list_folders ($identificator, $server, $prefix)
{
 echo "\n\n";
 echo "<H4>Lista dosarelor</H4>";
 echo "\nserver=$server, prefix=$prefix";
 $dosare = imap_listmailbox($identificator, $server, $prefix, "*");

 if ($dosare == false)
 {
 echo "\nFuncția imap_listmailbox() a eșuat."
 }
 else
 {
 foreach ($dosare as $cheie=>$valoare)
 {
 echo "\n$cheie=>$valoare";
 }
 }
}
```

Puteți apela această funcție folosind argumente asemănătoare cu următorul:

```
list_folders($identificator, "{localhost/imap:143}", "~/mail/")
```

\* În traducere afișează dosarele – N.T.

Argumentul \$identificator este, desigur, valoarea returnată de funcția care a deschis conexiunea IMAP. Așa cum s-a explicat anterior în acest modul, valoarea argumentului \$prefix trebuie să se conformeze politicii de denumire a dosarelor stabilite de administratorul IMAP.

## Crearea unui dosar

Iată cum se poate crea un nou dosar IMAP:

```
function create_folder*($identificator, $server, $prefix, $dosar);
{
 $dosar = $server . $prefix . "INBOX." . $dosar;
 echo "\n\n";
 echo "<H4>Creeaza dosarul $dosar</H4>";
 $ok = imap_createmailbox($identificator, $dosar);
 if ($ok === false)
 {
 echo "\nNu s-a putut crea dosarul.";
 print_error_stack();
 }
}
```

Cel de-al patrulea argument al acestei funcții specifică numele dosarului care va fi creat.

## Sugestie

Numele unui dosar IMAP trebuie să conțină numai litere, cifre și caractere de subliniere. Dacă doriți să creați un sub-dosar, puteți proceda astfel incluzând un punct în numele dosarului. Punctul se comportă ca separator de cale, analog caracterului slash folosit în căile din cadrul sistemului de fișiere.

## Modificarea numelui unui dosar

Iată o funcție definită de utilizator care modifică numele unui dosar:

```
function rename_folder*($identificator, $server, $prefix, $dosar_vechi, $dosar_nou);
{
 $dosar_vechi = $server . $prefix . "INBOX." . $dosar_vechi;
 $dosar_nou = $server . $prefix . "INBOX." . $dosar_nou;
 echo "\n\n";
 echo "<H4>Modifica numele $dosar_vechi in $dosar_nou</H4>";
 $ok = imap_renamemailbox($identificator, $dosar_vechi, $dosar_nou);
 if ($ok === false)
 {
 echo "\nNu a reusit sa modifice numele dosarului.";
 print_error_stack();
 }
}
```

\* În traducere creează dosarul – N.T.

\*\* În traducere modifică numele dosarului – N.T.

Așa cum s-a explicat anterior, poate fi necesară ajustarea modului de construire a numelui dosarului din numele gazdei serverului și din prefixul cutiei poștale.

## Ștergerea unui dosar

Iată o funcție care șterge un dosar IMAP:

```
function delete_folder*($identificator, $server,
 $prefix, $dosar)
{
 $dosar = $server . $prefix . "INBOX." . $dosar;
 echo "\n\n";
 echo "<H4>Sterge dosarul $dosar</H4>";
 $ok = imap_deletemailbox($identificator, $dosar);
 if ($ok === false)
 {
 echo "\nNu s-a putut crea dosarul.";
 print_error_stack();
 }
}
```

## Atenție

Spre deosebire de mesajele IMAP, care rămân în cutia poștală până când le ștergeți, un dosar IMAP șters este eliminat imediat și în mod irevocabil. Fiți atent atunci când scrieți programe care șterg dosare sau atunci când folosiți scripturi care conțin asemenea programe.

## Copierea mesajelor într-un dosar

IMAP poate copia un mesaj din dosarul curent într-un alt dosar. Iată o funcție definită de utilizator care execută această operație, dacă se cunoaște numărul mesajului IMAP și dosarul destinație:

```
function copy_message*($identificator, $prefix, $numar, $dosar)
{
 $dosar = $prefix . "INBOX." . $dosar;
 echo "\n\n";
 echo "<H4>Copiaza mesajul $numar in dosarul $dosar</H4>";
 $nr_mesaje = " " . $numar;
 $ok = imap_mail_copy($identificator, $nr_mesaje, $dosar);
 if ($ok === false)
 {
 echo "\nNu a fost copiat mesajul in dosarul specificat.";
 print_error_stack();
 }
}
```

Remarcați că această funcție nu preia un argument care specifică numele gazdei serverului. Deoarece serverele IMAP nu cooperează la copierea unui mesaj de la un

\* În traducere șterge dosarul – N.T.

\*\* În traducere copiază mesajul – N.T.

server la altul, nu este necesar – sau posibil – să se specifice numele gazdei serverului la copierea mesajelor IMAP.

## Mutarea mesajelor într-un dosar

Iată o funcție definită de utilizator care mută un mesaj IMAP din dosarul curent într-un alt dosar, fiind date numărul IMAP al mesajului și dosarul destinație:

```
function move_message*($identificator, $prefix, $numar, $dosar)
{
 $dosar = $prefix . "INBOX." . $dosar;
 echo "\n\n";
 echo "<H4>Muta mesajul $numar in dosarul $dosar</H4>";
 $nr_mesaj = "" . $numar;
 if ($ok === false)
 {
 echo "\nMesajul nu a fost mutat in dosarul specificat.";
 print_error_stack();
 }
}
```

Mesajele originale sunt doar marcate în vederea ștergerii și rămân în cutia poștală până la eliminarea efectivă.



### Sfatul specialistului

**Întrebare:** Acest modul a explicat modul de utilizare a numeroase funcții IMAP, dar există și alte funcții IMAP care mi-ar putea fi de folos?

**Răspuns:** Biblioteca IMAP furnizează multe alte funcții în afara celor descrise în acest modul. De exemplu, funcțiile `imap_search()` și `imap_scanmailbox()` vă permit să căutați mesaje care satisfac criterii specificate. De exemplu, puteți căuta mesaje al căror corp conține anumite texte. De asemenea, funcțiile IMAP folosite în acest modul furnizează frecvent opțiuni și caracteristici care nu au fost explicate în totalitate. Pentru mai multe informații privind biblioteca IMAP PHP, consultați manualul PHP pe suport electronic, la adresa [www.php.net](http://www.php.net).



### Test de evaluare

1. Care este protocolul folosit pentru expedierea mesajelor prin Internet?
2. În ce mod contribuie funcțiile definite de utilizator la simplificarea activității de programare?
3. Folosind funcția definită de utilizator adecvată descrisă în acest modul, scrieți o instrucțiune care copiază mesajul IMAP cu numărul 101 din dosarul curent

\* În traducere *mută mesajul* – N.T.

în dosarul "test". Se presupune că variabila `$mb` conține identificatorul asociat cu o conexiune IMAP deschisă, precum și că variabila `$pfx` conține prefixul cutiei poștale IMAP.

4. Folosind funcția definită de utilizator adecvată descrisă în acest modul, scrieți o instrucțiune care modifică numele dosarului "test1" în "test2". Se presupune că variabila `$mb` conține identificatorul asociat cu o conexiune IMAP deschisă, că variabila `$server` conține șirul server IMAP (care include parantezele acolade, numele gazdei serverului, protocolul și numărul portului), că variabila `$pfx` conține prefixul cutiei poștale IMAP, precum și că variabilele `$vechi`, respectiv `$nou` conțin numele dosarului.
5. Folosind funcția definită de utilizator adecvată descrisă în acest modul, scrieți o instrucțiune care afișează antetele asociate mesajului IMAP al cărui număr este dat de valoarea variabilei `$n`. Se va presupune că variabila `$mb` conține identificatorul asociat cu o conexiune IMAP deschisă.

# Modulul 13: Noțiuni fundamentale despre bazele de date și SQL

## Scopuri

- **Învățați care este modul de organizare a bazelor de date relaționale**
- **Învățați motivele pentru care bazele de date relaționale constituie medii mai bune de stocare a datelor decât fișierele**
- **Învățați să formați interogări SQL care obțin acces la datele relaționale și le manipulează**
- **Învățați să proiectați și să creați baze de date relaționale**

Acest modul explică bazele de date relaționale și modul de utilizare a acestora. În comparație cu fișierele, bazele de date relaționale prezintă multe avantaje, inclusiv o mai mare protecție a integrității datelor și asigurarea partajării datelor. Acest modul se concentrează asupra SQL, limbajul standard pentru crearea, accesul și manipulara bazelor de date relaționale. În cadrul modulului următor, veți învăța să încorporați instrucțiuni SQL în scripturile dumneavoastră PHP, astfel încât programele dumneavoastră PHP să poată lucra cu bazele de date relaționale. Conceptele explicate în acest modul se aplică majorității bazelor de date relaționale; cu toate acestea, detaliile – cu precădere sintaxa SQL – sunt cele referitoare la MySQL, cel mai popular limbaj de baze de date folosit cu PHP.

## Concepte ale bazelor de date relaționale

Nu cu mult timp în urmă, bazele de date relaționale constituiau o noutate. Pe atunci, alte categorii de baze de date, precum cele de rețea și ierarhice, erau „la modă”. Totuși, modelul bazelor de date relaționale s-a dovedit a fi mai eficient din punct de vedere al costurilor decât concurenții săi. Această secțiune explică modul de organizare a bazelor de date relaționale și rațiunile care justifică succesul modelului bazelor de date relaționale.

## Structura unei baze de date relaționale

O bază de date relaționale stochează datele în *tabele*, care amintesc de foile de calcul tabelar, iar fiecare tabel stochează informații despre un anumit tip de entitate. Practic, un tabel poate fi asimilat cu un fișier. De exemplu, o bază de date relațională aferentă unei edituri poate include tabele precum *carte* și *autor*.

Figura 13-1 prezintă un tabel caracteristic dintr-o bază de date relațională care prezintă angajații istorici ai Administrației Statelor Unite ale Americii. Primul rând al tabelului atribuie nume pentru fiecare coloană. Fiecare *rând* al tabelului, altul decât primul rând, descrie un singur angajat. De exemplu, al doilea rând descrie un angajat pe nume George Washington\*. Fiecare *coloană*, pe de altă parte, descrie un anumit *atribut* al angajatului. De exemplu, a doua coloană conține numele angajaților, iar a treia coloană conține anii în care s-au născut aceștia.

Pentru a se putea face referire la un anumit rând al tabelului, se obișnuiește ca fiecare tabel să conțină o coloană care identifică în mod unic fiecare rând. Această coloană se numește *cheia primară* a tabelului. În figura 13-1, coloana numită AngajatID servește drept cheie primară. Dacă nici o coloană nu conține o valoare unică pentru fiecare rând, se pot combina valorile mai multor coloane pentru a crea o *cheie primară compusă*.

Cheie primară

AngajatID	Nume	AnNastere
0001	George Washington	1732
0002	John Adams	1735
0003	Thomas Jefferson	1743

Tabel angajați

**Figura 13-1** Un tabel caracteristic dintr-o bază de date.

O bază de date *relațională* se numește astfel datorită capacității sale de a stabili relații între date din mai multe tabele. Figura 13-2 prezintă două tabele și o relație între acestea. Noul tabel conține informații despre meseriile caracteristice ale angajaților. Mai concret, tabelul îl identifică pe angajatul cel

mai priceput într-o anumită meserie. Numele meseriei servește drept cheie primară a tabelului, care mai conține, în afară de aceasta, o singură coloană.

Cheie primară

AngajatID	Nume	AnNastere
0001	George Washington	1732
0002	John Adams	1735
0003	Thomas Jefferson	1743

Tabel angajați

Cheie primară Cheie externă

Meserie	AngajatID
Arhitect	0003
General	0001
Filosof	0002

Tabel meserii

**Figura 13-2** O relație caracteristică între două tabele

\* General american (1732-1799), primul președinte al Statelor Unite ale Americii. – N.T.

Coloana respectivă stochează atributul `AngajatID` al angajatului care cunoaște meseria descrisă de un anumit rând. De exemplu, angajatul cu numărul 0003 este cel mai priceput arhitect. Rețineți că `AngajatID` este atât cheia primară a tabelului original, dar și o coloană din noul tabel. Coloana `AngajatID` a noului tabel se numește *cheie externă*; deși nu este cheia primară a noului tabel, este cheia primară a unui alt tabel.

Aplicația software care găzduiește o bază de date se numește *sistem de gestiune a bazelor de date* (SGBD). Există multe sisteme de gestiune a bazelor de date din surse deschise și comerciale. Printre cele mai populare asemenea sisteme se numără:

SGBD	Tip
DB2	Comercial
Interbase	În trecut comercial; în prezent din sursă deschisă
MySQL	Sursă deschisă
Oracle	Comercial
Postgresql	Sursă deschisă
SQL Server	Comercial
Sybase	Comercial

MySQL este cel mai popular sistem de gestiune a bazelor de date destinat utilizării cu PHP, în mare măsură deoarece este gratuit. Totuși, prin intermediul PHP este posibil accesul la aproape orice SGBD modern. Pentru aceasta, nu aveți nevoie decât de un program – cunoscut sub numele de *driver* – care se comportă ca o interfață între PHP și baza de date. Multe sisteme de gestiune a bazelor de date sunt asociate cu programe driver care se conformează standardului ODBC (Open Database Connectivity). Aceste sisteme de gestiune a bazelor de date sunt accesibile prin intermediul caracteristicii ODBC a limbajului PHP.

## Rațiuni de utilizare a bazelor de date relaționale

În comparație cu fișierele și bazele de date non-relaționale, bazele de date relaționale prezintă un număr de avantaje și câteva dezavantaje. Cunoscând atât avantajele, cât și dezavantajele, veți putea determina când este de preferat stocarea datelor într-un fișier, nu într-o bază de date.

## Facilitarea partajării datelor

Avantajul definitoriu al unui SGBD relațional îl constituie capacitatea de partajare a datelor. Acest fapt este important mai ales pentru aplicațiile bazate pe Web, deoarece mai mulți utilizatori pot obține acces la aceleași date aproape simultan. Sistemele de gestiune a bazelor de date relaționale includ elemente de protecție, proiectate pentru a preveni pierderea actualizărilor și deteriorarea datelor, care se pot produce în caz contrar în asemenea circumstanțe. Mai mult, sistemele de gestiune a bazelor de date au o arhitectură client-server care pune la dispoziția

utilizatorilor aflați la distanță, prin intermediul unei rețele, date stocate într-o locație centrală. Astfel, bazele de date relaționale furnizează partajarea datelor atât în timp, cât și în spațiu.

## Asigurarea independenței datelor

Independența datelor este un avantaj al bazelor de date care este depășit, ca importanță, numai de partajarea datelor. Când un program obține accesul la un fișier, datele sunt transferate programului în aceeași manieră în care sunt stocate. Prin contrast, programatorii folosesc un limbaj special pentru a solicita date dintr-o bază de date relațională. Programatorii pot solicita ca datele respective să fie transferate în orice formă o doresc aceștia, indiferent de modul de stocare a datelor. În particular, programatorii pot solicita numai coloanele unui tabel necesare într-o anumită aplicație.

Această caracteristică este importantă atunci când la o bază de date sunt adăugate coloane noi. Datorită independenței datelor, programele existente anterior continuă să funcționeze și după modificarea bazei de date. Prin contrast, adăugarea unui câmp la un fișier impune, în general, revizuirea fiecărui program care obține acces la fișier.

## Interogarea ad-hoc

Bazele de date relaționale înțeleg SQL (Structured Query Language\*), un limbaj relativ simplu, folosit pentru solicitarea datelor. Totuși, în ciuda simplității sale, SQL este un limbaj foarte puternic, care poate obține accesul la date stocate în mai multe tabele, poate filtra datele dorite și poate sorta, rezuma și afișa rezultatele.

În general, nu se pot anticipa toate modalitățile în care utilizatorii pot dori să obțină acces la date și să le vizualizeze. Ca atare, nu se pot scrie programe de aplicație care să satisfacă fiecare potențială necesitate de informații. Este aproape sigur că vor apărea unele cereri de date neprevăzute (sau ad hoc).

Utilizând SQL, este posibil accesul la datele stocate într-o bază de date relațională fără a scrie un program de aplicație, permițând frecvent evitarea întârzierilor și a costurilor implicate de programarea personalizată. Astfel, bazele de date relaționale permit satisfacerea tuturor cererilor ad-hoc de informații, care ar rămâne fără răspuns în alte situații.

## Organizarea datelor

În general, bazele de date relaționale își stochează datele într-un singur fișier sau catalog. Această caracteristică de organizare facilitează administrarea datelor, deoarece executarea copiei de siguranță, respectiv restaurarea unui singur fișier sau

\* În traducere *limbaj de interogare structurat* – N.T.



catalog sunt mai simplă decât operațiile similare aplicate unui set de fișiere stocat în mai multe cataloage.

## Asigurarea datelor

În general, bazele de date relaționale protejează datele împotriva accesului neautorizat. De exemplu, fișierele care stochează tabelele relaționale pot fi accesibile numai pentru administratorul de sistem și pentru un cont special de utilizator, creat pentru gestionarea bazei de date.

## Reducerea la minimum a experienței necesare în domeniul programării

În general, sistemele moderne de gestiune a bazelor de date folosesc complexitatea pentru a da iluzia simplității. Datorită complexității acestora, în general este mai simplu să se scrie un program care folosește o bază de date relațională decât să se scrie un program echivalent din punct de vedere funcțional, dar care folosește fișiere obișnuite. Mai mult, o aplicație scrisă folosind un SGBD va prezenta mai puține defecte decât o aplicație echivalentă din punct de vedere funcțional, scrisă folosind fișiere normale.

În general, autorii sistemelor de gestiune a bazelor de date beneficiază de o bogată experiență, pe care și-au utilizat-o prin crearea de programe reutilizabile la care alți programatori obțin acces prin intermediul funcțiilor definite cu interfețe simple. Așa cum un sistem de operare scutește programatorii de necesitatea de a înțelege mecanismele detaliate de funcționare ale dispozitivelor hardware, o bază de date relațională îi scutește pe programatori de necesitatea de a înțelege o varietate de probleme complexe care pot apărea la partajarea datelor.

## Obținerea eficienței în prelucrarea datelor

Datorită complexității lor, sistemele de gestiune a bazelor de date relaționale necesită mai multe cicluri de procesor pentru a satisface o cerere de date decât cele necesare pentru accesul la un fișier ordinar. În acest sens, sistemele de gestiune a bazelor de date relaționale sunt ineficiente. Totuși, dacă examinăm chestiunea dintr-o altă perspectivă, putem ajunge la o concluzie diferită.

De exemplu, doriți să calculați dimensiunea medie a gospodăriilor americane folosind datele biroului de recensământ. Dacă aceste date ar fi fost stocate într-un fișier obișnuit, ați scrie un program care să includă un ciclu care citește fiecare înregistrare a fișierului și incrementează contoare pentru dimensiune și numărul de gospodării. Să presupunem că fișierul de recensământ este stocat pe un calculator aflat la distanță, la care obțineți acces prin intermediul unei rețele. În acest caz,

fiecare înregistrare de recensământ este trimisă prin rețea, creând un adevărat blocaj de trafic. Totuși, dacă datele de recensământ ar fi fost stocate într-o bază de date relațională, puteți pur și simplu folosi SQL pentru a solicita calculul dimensiunii medii a unei gospodării. Astfel, singurele date trimise prin rețea ar fi rezultatul însuși. Deci, utilizarea unui SGBD relațional nu este întotdeauna mai puțin eficientă decât folosirea unor fișiere normale.

## Decizia de utilizare a unui SGBD relațional

Din punctul de vedere al unei firme, utilizarea unei tehnologii este adecvată atunci când avantajele utilizării depășesc costurile. În cazul unui SGBD relațional, principalul cost incremental în comparație cu fișierele obișnuite constă în necesitatea unor resurse mai mari de prelucrare a datelor. Aceasta presupune, desigur, că alegeți un SGBD din sursă deschisă, care este disponibil gratuit; în caz contrar, vor apărea costuri pentru achiziționarea și întreținerea sistemului comercial de gestiune a bazelor de date ales de dumneavoastră.

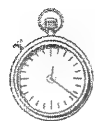
Așa cum s-a explicat în sub-sectiunile precedente, avantajele incrementale ale unui SGBD relațional sunt numeroase. Acolo unde acestea sunt importante, avantajele unui SGBD depășesc, în general, costurile. Fișierele normale rămân adecvate pentru date relativ statice, care nu sunt partajate, nu sunt supuse la interogări ad-hoc, nu sunt confidențiale sau extrem de valoroase și sunt folosite de un număr redus de programe. Cu alte cuvinte, implementarea unor aplicații foarte simple poate fi mai eficientă sub aspectul costurilor dacă se folosesc fișiere normale și nu SGBD; cu toate acestea, majoritatea aplicațiilor importante sunt mai eficiente din punct de vedere al costurilor dacă sunt implementate folosind un SGBD.



### Sfatul specialistului

**Întrebare:** Care este sistemul de gestiune a bazelor de date pe care trebuie să-l folosesc pentru aplicația mea?

**Răspuns:** Așa cum s-a menționat anterior, MySQL este cel mai important SGBD destinat utilizării cu PHP. Totuși, PostgreSQL este de asemenea un SGBD din sursă deschisă și este disponibil gratuit. În comparație cu MySQL, PostgreSQL furnizează funcții suplimentare, care facilitează scrierea programelor ce asigură integritatea tranzacțiilor. Trebuie să folosiți PostgreSQL dacă baza dumneavoastră de date va fi actualizată frecvent și cu volume mari de date. Dacă baza dumneavoastră de date este asociată unui sistem comercial, trebuie să aveți la dispoziție fonduri pentru achiziționarea unui SGBD comercial. Un SGBD comercial trebuie selecționat în funcție de numeroși factori, inclusiv experiența dumneavoastră cu producătorul respectiv și dimensiunea bugetului de care dispuneți.



### Test „la minut”

- Care este componenta unei baze de date relaționale care stochează informații despre o anumită categorie de entitate?
- Care este componenta unei baze de date relaționale care stochează informații despre un anumit exemplu de entitate?
- Care este componenta unei baze de date relaționale care stochează valorile unei anumite caracteristici pentru un set de entități?

## Implementarea unei baze de date

Implementarea unei baze de date relaționale este un subiect de o amploare considerabilă și a fost abordată în cadrul a numeroase cărți. Această secțiune oferă o trecere în revistă a implementării bazelor de date relaționale, care descrie procesele de proiectare și creare a unei baze de date pornind de la o perspectivă simplă, practică. Scopul secțiunii de față constă în a vă oferi cunoștințele necesare pentru a implementa baze de date MySQL simple, accesibile programelor PHP.

### Proiectarea unei baze de date

Un instrument frecvent utilizat de proiectare a bazelor de date constă din procesul cunoscut sub numele de *modelare entitate-relație* sau *modelare E-R*. În contextul modelării E-R, o *entitate* este similară cu un tabel relațional; cu alte cuvinte, conține date care descriu un set de individualități corelate. Modelarea E-R este un proces în cadrul căruia coloanele, entitățile și relațiile între entități sunt descoperite și organizate. Un model E-R poate fi rafinat cu ușurință, pentru a genera o structură a unei baze de date, care poate fi transformată într-o bază de date relațională efectivă.

### Modelare E-R

Procesul de modelare E-R constă din patru faze principale:

1. Identificarea coloanelor
2. Gruparea coloanelor în entități
3. Identificarea cheilor primare
4. Identificarea cheilor externe

#### Răspunsuri la test

- Tabel
- Rând
- Coloană

### Identificarea coloanelor

Prima operație din cadrul procesului de modelare E-R este identificarea coloanelor. Deseori, această operație este executată de un grup de persoane, care acționează sub îndrumarea și sfatul unei persoane cu experiență în domeniu.

Să ne reamintim că o coloană înregistrează o singură caracteristică a unei entități. În esență, grupul identifică posibile coloane punând întrebarea: „Care sunt datele sau caracteristicile pe care trebuie să le stocheze sistemul?”. Coloanele candidate sunt puse pe listă de îndată ce sunt identificate. În acest scop se folosește frecvent o tablă, deci participanții pot vedea lista pe măsură ce aceasta începe să se contureze și pot modifica lista rapid, conform necesităților.

În încercarea de identificare a coloanelor, în general este util să se răspundă la unele întrebări conexe, cum sunt următoarele:

- Care sunt deciziile pe care sistemul trebuie să le ia sau să le susțină?
- Care sunt operațiile pe care sistemul trebuie să le execute sau să le susțină?
- Care sunt datele necesare pentru a lua aceste decizii și pentru a efectua aceste operații?

În momentul în care nu mai pot fi găsite și alte coloane candidate, procesul trece la faza următoare, și anume gruparea coloanelor în entități.

### Gruparea coloanelor în entități

De obicei, este evident că unele coloane sunt corelate, în sensul că fac referire la un anumit set de individualități corelate. De exemplu, coloane precum *autor*, *titlu* și *pret de coperta* se pot corela cu noțiunea de *cărți*. Ca atare, aceste coloane pot fi grupate pentru a forma o entitate, cum este *carte*. Uneori, o coloană dată este corelată cu mai multe entități; în acest caz, coloana poate apărea de mai multe ori pe listă.

O dată entitățile identificate, este util să acordăm o oarecare atenție numelor. Limbajul SQL folosit cu bazele de date relaționale impune unele restricții asupra numelor. Este utilă revizuirea numelor care nu se conformează acestor restricții, pentru a evita problemele ce pot apărea în etapele viitoare ale procesului de proiectare. Cel mai bine este ca numele coloanelor și ale entităților să respecte următoarele restricții:

- Trebuie să înceapă cu o literă
- Trebuie să conțină numai litere, cifre și caracterul de subliniere (`_`).
- Lungimea lor nu trebuie să depășească 64 de caractere
- Trebuie să fie tratate ca insensibile la diferența între majuscule și minuscule (de exemplu, nu trebuie să aveți coloane distincte cu numele *abc* și *ABC*)

După ce ați grupat coloanele în entități, puteți trece la identificarea cheii primare pentru fiecare entitate.

## Notă

Majoritatea sistemelor de gestiune a bazelor de date, inclusiv MySQL, impun restricții mai puțin severe decât cele recomandate. Dar, prin respectarea restricțiilor recomandate, puteți evita o mulțime de probleme cu SGBD, HTML și PHP.

## Identificarea cheilor primare

În cele din urmă, fiecare entitate va deveni un tabel relațional și, ca atare, va trebui să aibă o cheie primară. Examinați fiecare entitate pentru a determina dacă una dintre coloanele sale asociate are o valoare unică pentru fiecare dintre aparițiile entității. Dacă o asemenea coloană există, o veți identifica drept cheie primară a entității. De exemplu, puteți identifica valoarea `CodNumericPersonal` ca fiind cheia primară a unei entități care conține informații referitoare la contribuabili pentru anul în curs.

Puteți găsi unele entități care nu conțin nici o coloană adecvată pentru rolul de cheie primară. Într-o asemenea situație, puteți căuta o serie de coloane care au o valoare combinată unică. Dacă descoperiți o asemenea serie, o puteți identifica drept cheie primară compusă a entității. De exemplu `CodNumericPersonal` și `AnFiscal` pot servi împreună drept cheie primară a unei entități care conține informații referitoare la contribuabili pentru mai mulți ani.

S-ar putea să nu descoperiți nici o coloană sau serie de coloane care să identifice în mod unic fiecare apariție a unei entități. În acest caz, creați o coloană nouă, care va conține o identificare artificială unică, și identificați noua coloană ca fiind cheia primară a entității. De exemplu, în cazul unei entități numite angajat, puteți denumi identificarea artificială `angajatid` sau `angajatnr`, ultimul nume fiind o abreviere frecvent folosită pentru o coloană care altfel s-ar fi numit `angajat_numar`.

## Sugestie

Puteți dori să folosiți o identificare artificială unică chiar și atunci când una sau mai multe coloane pot servi drept cheie primară. Astfel, evitați problemele care apar când identificatori presupuși unici se dovedesc a nu fi unici. De exemplu, se presupune că valoarea codului numeric personal este unică; dar un angajat poate introduce informații incorecte, determinând un conflict între identificatorul propriu presupus unic și identificatorul unui alt angajat.

## Identificarea cheilor externe

Operația finală și cea mai dificilă din cadrul activității de modelare E-R o constituie identificarea cheilor externe. Să ne reamintim că acestea sunt pur și simplu coloane ale căror valori sunt corelate cu acelea ale valorilor cheilor primare ale unei entități oarecare. Procesul de identificare a cheilor externe constă în compararea coloanelor cu cheile primare și, pentru fiecare combinație posibilă, în răspunsul la întrebarea: „Există o relație între valoarea acestei coloane și valoarea acestei chei primare?”

Majoritatea celor care practică modelarea E-R folosesc un fel de diagramă, cunoscută sub numele de diagramă E-R, pentru a le fi de ajutor la documentarea cheilor externe. Figura 13-3 prezintă o diagramă E-R caracteristică, diagramă care descrie tabelele relaționale prezentate anterior în figura 13-2. O diagramă E-R reprezintă entitățile sub formă de dreptunghiuri, iar relațiile dintre entități sub formă de romburi. O relație există oriunde a fost identificată o cheie externă. Relația stie meseria din figura 13-3 s-a stabilit între entitățile numite angajat și meserii. Uneori, diagramele E-R prezintă câmpurile asociate fiecărei entități; deoarece astfel se obțin deseori diagrame aglomerate, acest procedeu nu este frecvent folosit. Cu toate acestea, îl puteți găsi util, mai ales pentru modelele E-R mici.

*Stricto sensu*, această activitate de modelare E-R implică mai mult decât o simplă identificare a cheilor externe. O dată identificată o relație, aceasta trebuie clasificată și eventual revizuită. Pentru a clasifica relația, gândiți-vă la numărul de apariții ale entității implicate în relație, care poate fi zero, unu sau mai multe. În relația stie meseria, fiecare angajat are exact o meserie. O asemenea relație se numește *relație de tip 1:1*. Totuși, sunt posibile și alte *cardinalități* ale relațiilor, așa cum se numesc acestea.

De exemplu, o carte poate avea mai mulți autori. Astfel, relația dintre entitățile numite carte și autor este o relație de tip unu la mai mulți și se abreviază frecvent sub forma 1:N. Unele relații sunt opționale; de exemplu, un angajat poate fi căsătorit sau nu. Relația dintre angajat și sot/sotie este o relație de tip 1:0. Cu alte cuvinte, un angajat poate fi căsătorit sau nu; dar un angajat căsătorit are exact un soț, respectiv o soție.

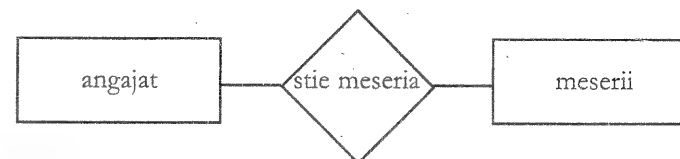


Figura 13-3: O diagramă E-R caracteristică.

Și mai interesante sunt relațiile de tip N:N. Un exemplu de asemenea relație este cea între curs și student. Relația este de tip N:N deoarece la fiecare curs pot fi înscriși mai mulți studenți, iar fiecare student poate fi înscris la mai multe cursuri. O asemenea relație este nedefinită și indică lipsa unei entități, în acest caz, entitatea înscriere. Ori de câte ori descoperiți o entitate N:N, trebuie să determinați și să adăugați entitatea care lipsește. După ce ați adăugat entitatea lipsă, trebuie să modificați relațiile.

Deseori, entitatea lipsă este corelată cu una sau mai multe coloane care lipsesc. De exemplu, în cazul entității lipsă înscriere, coloana nota va lipsi, deoarece nu poate fi plasată în mod justificat nici în tabelul curs, nici în tabelul student. Coloana

nota se refera la o relație între un curs și un student, nu numai la un curs sau la un student.

După adăugarea entităților care lipsesc, toate relațiile de tip N:N trebuie să dispară. De exemplu, relația între curs și înscriere este de tip 1:N, deoarece pot exista mai multe înscrieri la un curs dat, dar fiecare înscriere se referă la un anumit curs. Similar, relația dintre înscriere și student este N:1, deoarece fiecare înscriere se referă la un anumit student, care se poate înscrie la mai multe cursuri.

După ce ați eliminat relațiile de tip N:N, puteți lua în considerare normalizarea bazei de date descrise de modelul E-R.

## Normalizarea unei baze de date

O bază de date normalizată este una care a fost transformată astfel încât să satisfacă o serie de reguli. Regulile de normalizare a bazelor de date sunt descrise ca proprietăți pe care o bază de date care respectă aceste reguli trebuie să le aibă, proprietăți cunoscute sub numele de forme. Setul cel mai frecvent aplicat de reguli de normalizare a bazelor de date include trei reguli, care descriu prima, a doua și a treia formă normală.

Aceste forme sunt destinate a preveni problemele care pot apărea în cadrul bazelor de date care nu le respectă. Totuși, aceste reguli sunt derutante și dificil de înțeles pentru mulți. Sub-sețiunea de față prezintă pe scurt o abordare de bun simț a normalizării bazelor de date, adecvată pentru evitarea a numeroase probleme frecvent întâlnite legate de proiectarea bazelor de date. Proiectanții bazelor de date foarte simple pot opta în mod logic pentru omiterea în totalitate a normalizării bazelor de date, întrucât problemele de proiectare pot fi corectate pur și simplu la apariția lor, în cursul programării sau al utilizării. Totuși, normalizarea bazelor de date este esențială pentru bazele de date mari, unde costul și efortul de descoperire și remediere a unei erori poate depăși semnificativ costul și efortul implicate în normalizarea bazelor de date.

### Regula 1: Este permisă numai utilizarea valorilor atomice

Un tabel dintr-o bază de date trebuie să conțină numai valori atomice. Cu alte cuvinte, nici o coloană nu trebuie să conțină valori compuse. De asemenea, nici o coloană nu trebuie să reprezinte un grup repetitiv.

Această regulă se aplică pentru coloane precum `nume`, alcătuită din prenume, inițiala mijlocie și numele de familie. O asemenea coloană are o valoare compusă și trebuie divizată în trei coloane separate: `prenume`, `initiala_mijlocie` și `nume_familie`. Această regulă este frecvent încălcată, deseori la un preț redus. Dezavantajul încălcării acestei reguli este îngreunarea în SQL a accesului la componentele unei coloane compuse.

Interdicția îndreptată împotriva grupurilor care se repetă este o problemă mai serioasă. Să luăm în considerare un tabel care include un grup repetitiv de adrese de e-mail. Care este numărul de repetiții permis? Două, trei, cinci, zece? Pentru a evita limitarea artificială a numărului de repetiții, structura bazei de date trebuie să fie revizuită, pentru a plasa grupul repetitiv într-un alt tabel. De exemplu, structura următoare:

```
contact table:
contactid (cheie primara)
nume
email1
email2
email3
```

trebuie înlocuită cu o structură ca următoarea:

```
contact table:
contactid (cheie primara)
nume

email table:
emailid (cheie primara)
contactid (cheie externa)
email
```

### Sugestie

Pentru a evita încălcarea acestei reguli, eliminați toate grupurile repetitive prin definirea unuia sau mai multor tabele care să conțină grupurile respective.

### Regula 2: Bazați-vă pe cheia primară în totalitatea sa

Coloanele dintr-un tabel trebuie să se bazeze pe cheia primară în totalitatea sa. Pentru a vedea care este modul de funcționare al acestei reguli, luați în considerare următorul tabel, care încalcă regula:

```
inscriere table:
studentid
cursid
studentnume
```

Să presupunem că `studentid` și `cursid` au fost desemnate drept cheie primară compusă. Coloana `studentnume` este asociată unui student, nu unei înscrieri. În consecință, depinde de valoarea `studentid`, dar nu și de valoarea `cursid`. Dacă un student s-a înscris la mai multe cursuri, atunci studentului respectiv îi vor fi asociate mai multe rânduri înscriere, fiecare cu o coloană `studentnume`. Dacă se descoperă că numele studentului a fost greșit ortografiat, trebuie corectate mai multe rânduri din tabel; în caz contrar, unele rânduri vor avea valori incorecte în anumite coloane. Se vor evita bazele de date inconsecvente, deci structurile în care o coloană depinde numai de o porțiune a cheii primare sunt interzise prin această regulă.

### Sugestie

Această regulă se aplică numai tabelelor cu o cheie primară compusă. O metodă bună de a evita încălcarea acestei reguli este a folosi o cheie primară artificială și nu o cheie primară compusă, convenabilă în alte condiții.

### Regula 3: Bazați-vă numai pe cheia primară

Coloanele dintr-un tabel trebuie să se bazeze numai pe cheia primară. Iată un exemplu de tabel care încalcă regula:

```
curs table:
 cursid (cheie primara)
 cursnume
 deptid
 deptnume
```

Acest tabel înregistrează identificatorul cursului, numele cursului, identificatorul departamentului și numele departamentului pentru cursuri. Cu toate acestea, coloana deptnume nu depinde de cursid; în schimb, depinde de deptid. Cu alte cuvinte, tabelul descrie atât cursurile, cât și departamentele. În consecință, numele departamentului apare în mod redundant în fiecare rând care se referă la un curs asociat departamentului. Structura trebuie revizuită după cum urmează:

```
curs table:
 cursid (cheie primara)
 cursnume
 deptid (cheie externa)

departament table:
 deptid (cheie primara)
 deptnume
```

### Sugestie

Pentru a evita încălcarea acestei reguli, căutați tabele care conțin informații despre mai multe categorii de entități. Toate aceste tabele trebuie divizate în tabele separate, unite printr-o cheie externă.

### Rafinarea modelelor E-R

Ultima operație de finețe aplicată unui model E-R constă în specificarea unui tip de date pentru fiecare coloană. Majoritatea bazelor de date relaționale acceptă următoarele tipuri de date generale:

- Caracter
- Întreg
- Zecimal
- Dată și oră
- Binar

Tabelul 13-1 rezumă numeroase tipuri de date frecvent utilizate, acceptate de MySQL și de majoritatea celorlalte sisteme de baze de date relaționale. Totuși, MySQL acceptă multe alte tipuri de date. Consultați manualul de referință MySQL pentru mai multe informații cu privire la aceste tipuri și la altele.

**Tabelul 13-1** Principalele tipuri de date din MySQL

Tip de date	Descriere
BLOB	Date binare arbitrare, cu o lungime maximă de 65535 octeți.
CHAR( <i>m</i> )	Un șir de caractere de lungime fixă, cu un maxim de <i>m</i> caractere, unde <i>m</i> este mai mic decât 256. Pentru obținerea lungimii dorite, se inserează spații finale.
DATE	O dată în format an-lună-zi; de exemplu 2005-12-31.
DECIMAL	Un număr zecimal, reprezentat sub formă de șir cu <i>m</i> cifre, din care <i>d</i> se află la dreapta punctului zecimal. Dacă <i>m</i> și <i>d</i> sunt omise, în mod prestabilit se vor utiliza valorile 10 și 0.
DECIMAL( <i>m</i> , <i>d</i> )	
DOUBLE	Un număr cu virgulă mobilă, cu dublă precizie, având o lățime de afișare egală cu <i>m</i> și un număr de <i>d</i> cifre după virgulă.
DOUBLE( <i>m</i> , <i>d</i> )	
FLOAT( <i>m</i> , <i>d</i> )	Un număr cu virgulă mobilă, cu simplă precizie, având o lățime de afișare egală cu <i>m</i> și un număr de <i>d</i> cifre după virgulă.
INTEGER	Un întreg pe 32 de biți. Dacă se specifică atributul UNSIGNED, domeniul de valori este cuprins între 0 și 4294967295; în caz contrar, domeniul este cuprins între valorile -2147483648 și 2147483647.
INTEGER UNSIGNED	
NUMERIC	Similar cu DECIMAL.
NUMERIC( <i>m</i> , <i>d</i> )	
REAL	Similar cu DOUBLE.
REAL( <i>m</i> , <i>d</i> )	
SMALLINT	Un întreg pe 16 biți. Dacă se specifică atributul UNSIGNED, domeniul de valori este cuprins între 0 și 65535; în caz contrar, domeniul este cuprins între valorile -32768 și 32767.
SMALLINT UNSIGNED	
TIME	Ora în format oră-minut-secundă; de exemplu, 08-30-00.
TIMESTAMP	O valoare de tip dată și oră, în format an-lună-zi oră-minut-secundă; de exemplu, 1970-01-01 00:00:00. Această reprezentare este similară celei returnate de funcțiile UNIX și nu poate codifica date situate dincolo de un anumit moment al anului 2037.
TIMESTAMP( <i>m</i> )	
VARCHAR( <i>m</i> )	Un șir caracter de lungime variabilă, cu un maximum de <i>m</i> caractere, unde <i>m</i> este mai mic decât 256. Spațiile finale au fost eliminate.

Iată unele reguli empirice pentru selectarea tipurilor de date:

- Alegeți BLOB ca tip pentru datele pe care nu trebuie să le manipulați și la care nu veți obține acces prin intermediul limbajului SQL.
- Alegeți un tip dată sau oră adecvat pentru coloanele care conțin date calendaristice sau ore.
- Alegeți un tip numeric pentru coloanele folosite în calcule.
  - Pentru cantități foarte mari sau foarte mici, alegeți DOUBLE ca tip de date.

- Pentru coloane care conțin numere fără parte zecimală de dimensiuni adecvate, alegeți `SMALLINT` sau `INTEGER` ca tip de date.
- Pentru alte coloane care conțin date numerice, alegeți `DECIMAL` ca tip de date.
- Alegeți `CHAR` sau `VARCHAR` ca tip pentru celelalte coloane, chiar și pentru cele care conțin mai ales cifre, cum ar fi un cod poștal.

### Sugestie

Când alegeți un tip de date, nu uitați să alocați spațiu pentru eventuale creșteri. De exemplu, nu specificați un număr de client format din două cifre decât dacă sunteți sigur că nu veți avea niciodată mai mult de 100 de clienți.

## Crearea unei baze de date MySQL

Administratorul de sistem creează baze de date MySQL. La început, o bază de date nu conține tabele. Pentru a crea un tabel într-o bază de date, folosiți un sub-limbaj SQL special, cunoscut sub numele de *Data Definition Language*\* (DDL). Această sub-sectiune este dedicată formelor pe care le pot lua comenzile DDL. Puteți emite comenzi DDL și alte comenzi SQL prin intermediul unui interpretor SQL sau prin intermediul PHP. Proiectele din acest modul prezintă modul de emitere a comenzilor SQL folosind un interpretor SQL. Modulul următor prezintă modul de emitere a comenzilor SQL utilizând PHP.

Pentru a crea un tabel într-o bază de date, emiteți comanda `CREATE TABLE`, care are următoarea formă:

```
CREATE TABLE tabel (coloana tip, coloana tip, ...);
```

unde *tabel* este numele tabelului, *coloana* este numele unei coloane, *tip* este tipul datelor incluse în coloană, iar ... arată că se poate specifica un număr nedefinit de coloane și tipuri. De exemplu, comanda următoare creează un tabel numit *carte*, care conține coloanele ISBN (un identificator unic asociat unei cărți), titlu și pret:

```
CREATE TABLE carte (carteid CHAR(10), titlu VARCHAR(255), pret DECIMAL(5,2));**
```

În general, SQL nu este sensibil la diferența între majuscule și minuscule. Deci, dacă preferați, puteți emite comanda următoare, care se comportă exact la fel ca și precedenta:

```
create table carte (carteid char(10), titlu varchar(255), pret decimal(5,2));
```

\* În traducere *limbaj de definiție a datelor* – N.T.

\*\* S-a preferat acest tip de date deoarece prețurile, în Statele Unite, sunt exprimate în dolari și cenți. Pentru a exprima un preț de carte în România, se poate folosi definiția `INTEGER UNSIGNED`, deoarece nu se mai folosește banul ca subdiviziune a leului. – N.T.

### Sugestie

Programele dumneavoastră SQL vor fi mai ușor de citit dacă respectați un anumit stil. De exemplu, puteți scrie toate cuvintele cheie SQL cu majuscule, iar cuvintele furnizate de programator – cu minuscule.

În afara tipului de date, puteți specifica numeroase atribute opționale ale unei coloane:

Atribut	Descriere
NOT NULL	Fiecare rând trebuie să conțină o valoare a coloanei asociate; valorile nule nu sunt permise.
DEFAULT <i>valoare</i>	Dacă nu este dată o valoare a coloanei asociate, se va presupune valoarea specificată.
AUTO-INCREMENT	MySQL va repartiza în mod automat un număr de serie ca valoare a coloanei asociate.
PRIMARY KEY	Coloana asociată este cheia primară a tabelului care o conține.

Iată o comandă `CREATE TABLE` ceva mai complicată, care folosește unele atribute opționale:

```
CREATE TABLE carte (carteid CHAR(10) PRIMARY KEY,
titlu VARCHAR(255) NOT NULL,
pret DECIMAL(5,2) DEFAULT 50.00);
```

### Ștergerea unui tabel

Ștergerea unui tabel este o operație simplă. Prin ștergerea unui tabel, sunt eliminate toate rândurile incluse în tabel. Pentru a șterge un tabel, emiteți următoarea comandă:

```
DROP TABLE tabel;
```

unde *tabel* este numele tabelului care urmează a fi șters.

### Atenție

Ștergerea unui tabel este un act irevocabil; asigurați-vă că intenționați să ștergeți tabelul înainte de a emite o comandă `DROP TABLE`; de asemenea, asigurați-vă că ați scris corect comanda înainte de a apăsa pe tasta ENTER.

### Modificarea unui tabel

După crearea unui tabel, îl puteți modifica prin emiterea unei comenzi `ALTER TABLE`. O formă a comenzii vă permite să ștergeți o coloană din tabel:

```
ALTER TABLE tabel DROP coloana;
```

unde *tabel* este numele tabelului care va fi modificat, iar *coloana* este numele coloanei care va fi ștearsă. De exemplu, pentru a șterge coloana *pret* din tabelul *carte*, emiteți comanda

```
ALTER TABLE carte DROP pret;
```



## Atenție

Ștergerea unei coloane este un act irevocabil; asigurați-vă că intenționați să ștergeți coloana înainte de a emite o comandă ALTER TABLE; de asemenea, asigurați-vă că ați scris corect comanda înainte de a apăsa pe tasta ENTER.

O altă formă a comenzii vă permite să adăugați o nouă coloană în tabel:

```
ALTER TABLE tabel ADD coloana tip [opțiuni];
```

unde *tabel* este numele tabelului care va fi modificat, *coloana* este numele coloanei care va fi adăugată, *tip* este tipul noii coloane, iar *[opțiuni]* constituie toate opțiunile dorite, precum PRIMARY KEY. De exemplu, pentru a adăuga din nou coloana pret la tabelul carte, emiteți comanda:

```
ALTER TABLE carte ADD pret DECIMAL(5,2) DEFAULT 50.00;
```

## Acordarea și revocarea privilegiilor de acces

Când un utilizator încearcă să obțină acces la o bază de date relațională, SGBD verifică dacă utilizatorul are permisiunea de a executa operația. Administratorul de sistem poate folosi comanda MySQL GRANT pentru a autoriza un utilizator să obțină accesul la un tabel din baza de date. Comanda are următoarea formă:

```
GRANT ALL ON tabel TO utilizator IDENTIFIED BY 'parola';
```

unde *tabel* este numele tabelului, *utilizator* este numele contului de utilizator, iar *parola* este parola pe care o va furniza utilizatorul pentru a-și proba identitatea. Alternativ, administratorul de sistem poate autoriza un utilizator să obțină acces la orice tabel dintr-o bază de date specificată, folosind următoarea formă a comenzii GRANT:

```
GRANT ALL ON baza_de_date.* TO utilizator IDENTIFIED BY 'parola';
```

De exemplu, comanda următoare autorizează pe utilizatorul php să obțină acces la toate tabelele din baza de date numită testdb, ori de câte ori utilizatorul furnizează parola specificată:

```
GRANT ALL ON testdb.* TO php IDENTIFIED BY 'eusuntala';
```

Comanda REVOKE se poate folosi pentru retragerea privilegiilor acordate anterior. Comanda are următoarele forme:

```
REVOKE ALL ON tabel FROM utilizator;
REVOKE ALL ON baza_de_date.* FROM utilizator;
```

De exemplu, pentru a revoca toate privilegiile utilizatorului baiat\_rau, emiteți comanda:

```
REVOKE ALL ON *.* FROM baiat_rau;
```

## ? Sfatul specialistului

**Întrebare:** Așa cum s-a arătat anterior, accesul la un tabel relațional pare a fi ceva de genul „totul sau nimic”. Nu există nici o modalitate de a se acorda acces numai la anumite coloane?

**Răspuns:** Da, administratorul de sistem poate folosi o formă mai complexă a comenzii GRANT pentru a autoriza accesul numai la coloanele specificate. Forma corespunzătoare a comenzii este:

```
GRANT privilegiu (coloane) ON tabel
TO utilizator IDENTIFIED BY 'parola';
```

sau

```
GRANT privilegiu (coloane) ON baza_de_date.*
TO utilizator IDENTIFIED BY 'parola';
```

unde *privilegiu* este privilegiul care urmează a fi extins, *coloane* sunt coloanele cărora li se aplică privilegiul, iar *tabel*, *baza\_de\_date*, *utilizator* și *parola* au semnificațiile cunoscute.

Sunt permise și forme similare ale comenzii REVOKE:

```
REVOKE privilegiu (coloane) ON tabel
FROM utilizator;
```

sau

```
REVOKE privilegiu (coloane) ON baza_de_date.*
FROM utilizator;
```

Între privilegiile posibile se numără următoarele:

- INSERT, care permite inserția rândurilor care conțin coloana specificată
- SELECT, care permite accesul la rândurile care conțin coloana specificată
- UPDATE, care permite actualizarea rândurilor care conțin coloana specificată

De exemplu, pentru a permite unui utilizator să obțină accesul la o coloană, fără a o modifica, puteți folosi o secvență de comenzi similară cu următoarea:

```
REVOKE ALL ON carte FROM php;
GRANT
SELECT(carteid, titlu, pret),
INSERT(carteid, titlu, pret),
UPDATE(carteid, titlu, pret)
ON carte TO php IDENTIFIED BY 'eusuntala';
REVOKE INSERT(pret) ON carte FROM php;
REVOKE UPDATE(pret) ON carte FROM php;
```

Rețineți că prima comandă revocă toate privilegiile de la nivelul tabelului; în caz contrar, aceste privilegii le vor elimina pe cele situate la nivel de coloană.

Caracteristica privilegiilor furnizată de MySQL este extrem de sofisticată și furnizează mult mai multe opțiuni. Pentru mai multe informații, consultați manualul SQL pe suport electronic, de la adresa [www.mysql.com](http://www.mysql.com).



### Test „la minut”

- Care este numele tehnicii de modelare frecvent folosită în proiectarea bazelor de date?
- Care este cardinalitatea relațiilor care trebuie să fie înlocuite la proiectarea unei baze de date?
- Care este numele procesului de adaptare a unei baze de date la o serie de reguli destinate a preveni apariția erorilor comune de proiectare?
- Care este numele sub-limbajului SQL folosit pentru crearea bazelor de date?

## Accesul la datele dintr-o bază de date: interogările SQL

În afară de Data Definition Language, SQL include Data Manipulation Language\* (DML). DML vă permite să formați interogări, care obțin accesul la datele stocate într-o bază de date relațională și raportează aceste date. De asemenea, puteți folosi DML pentru a insera, actualiza și șterge rândurile dintr-un tabel. Celelalte secțiuni ale acestui modul vor trata despre DML, iar secțiunea de față va aborda cea mai elementară formă de interogare: comanda SELECT simplă.

Cea mai simplă interogare posibilă raportează toate coloanele din toate rândurile unui tabel. Interogarea are următoarea formă:

```
SELECT * FROM tabel;
```

unde *tabel* este numele tabelului la care se va obține accesul. Formatul datelor de ieșire plasează fiecare rând al tabelului pe o linie separată și prezintă coloanele într-o ordine arbitrară. Datele de ieșire includ numele coloanelor și caractere simulate de desenare a liniilor, care separă coloanele. De exemplu, rulând această interogare asupra tabelului angajat se produc date de ieșire similare cu următoarele:

```
+-----+-----+
| angajatnr | nume |
+-----+-----+
| 1 | George Washington |
| 3 | T. Jefferson |
+-----+-----+
2 rows in set (0.00 sec)
```

#### Răspunsuri la test:

- Modelare E-R
- N:N
- Normalizarea bazelor de date
- Data Definition Language

\* În traducere *limbaj de manipulare a datelor*. – N.T.

Dacă doriți să selectați numai anumite coloane sau să raportați coloanele într-o anumită ordine, puteți folosi următoarea formă alternativă a comenzii SELECT:

```
SELECT coloana1, coloana2 FROM tabel;
```

unde *tabel* este numele tabelului, iar *coloana1* și *coloana2* sunt coloanele la care se va obține accesul și al căror conținut va fi raportat. Puteți specifica una, două sau mai multe coloane; pur și simplu separați numele fiecărei coloane de vecinii săi prin intermediul unei virgule. De exemplu, iată o interogare care inversează ordinea coloanelor în comparație cu interogarea anterioară:

```
SELECT nume, angajatnr FROM angajat;
```

În continuare, sunt prezentate datele de ieșire caracteristice ale acestei interogări:

```
+-----+-----+
| nume | angajatnr |
+-----+-----+
| George Washington | 1 |
| T. Jefferson | 3 |
+-----+-----+
2 rows in set (0.00 sec)
```

Deosebi, este necesară numai raportarea acelor rânduri care satisfac un anumit criteriu. Clauza WHERE vă permite să specificați o condiție; rândurile care nu satisfac condiția nu sunt raportate. De exemplu, iată o interogare care raportează un singur rând:

```
SELECT angajatnr, nume FROM angajat
WHERE angajatnr=1;
```

Forma condițiilor folosite în sub-limbajul DML al limbajului SQL este similară cu aceea a condițiilor PHP. Puteți folosi oricare din următorii operatori relaționali:

```
= Egalitate
<> Inegalitate
!= Inegalitate
< Mai mic decât
> Mai mare decât
<= Mai mic sau egal cu
=> Mai mare sau egal cu
```

Puteți compara valoarea unei coloane cu aceea a unei alte coloane, respectiv valoarea unei coloane cu o valoare literală. Valorile literale șir SQL trebuie să fie incluse între ghilimele simple, nu între ghilimelele duble permise de PHP.

### Sugestie

Când unei coloane nu i-a fost repartizată nici o valoare, SQL îi atribuie valoarea specială NULL. De asemenea, programatorii pot atribui în mod explicit valoarea NULL unei coloane. Comparațiile obișnuite cu valori NULL, care folosesc operatorii de (in)egalitate, vor returna un rezultat fals. Totuși, puteți folosi operatorul special <=>, care compară valorile ținând cont de valoarea NULL. Dacă folosiți acest operator pentru a compara două valori NULL, se obține un rezultat adevărat.

De asemenea, SQL include numeroși operatori de comparație non-algebrici:

x BETWEEN y AND z	Adevărat, dacă valoarea lui x este cuprinsă între valorile lui y și z.
x LIKE y	Adevărat dacă valoarea lui x este echivalentă cu modelul y.
x NOT LIKE y	Adevărat dacă valoarea lui x nu este echivalentă cu modelul y.
x IN (y <sub>1</sub> , y <sub>2</sub> )	Adevărat dacă valoarea lui x este un membru al listei y <sub>1</sub> , y <sub>2</sub> . Lista poate conține unul, doi sau mai mulți membri.
x NOT IN (y <sub>1</sub> , y <sub>2</sub> )	Adevărat dacă valoarea lui x nu este un membru al listei y <sub>1</sub> , y <sub>2</sub> . Lista poate conține unul, doi sau mai mulți membri.
x IS NULL	Adevărat dacă x are valoarea NULL.
x IS NOT NULL	Adevărat dacă x nu are valoarea NULL.

Sub-limbajul folosit pentru specificarea modelelor asociate operatorului LIKE este diferit de cel folosit de PHP sau de shell-ul UNIX. Meta-caracterul % corespunde unui număr de zero sau mai multe caractere, iar meta-caracterul \_ corespunde unui singur caracter. Modelele, ca și șirurile, sunt incluse între ghilimele simple. De exemplu, modelul '%ar%' corespunde oricărui șir care conține sub-șirul 'ar', inclusiv șiruri precum 'ar', 'arc' și 'un zar'.

### Sugestie

Pentru a plasa un caracter % sau \_ într-un șir test, folosiți secvența \% sau \\_; ca în PHP, caracterul slash orientat înapoi determină interpretarea celor două caractere ca având semnificația lor literală, nu ca meta-caractere.

Ca și PHP, sub-limbajul DML din SQL vă permite să formați expresii logice care combină expresiile relaționale. Puteți folosi oricare din următorii operatori logici:

AND	ȘI, adevărat dacă ambii operanzi sunt adevărați
OR	SAU inclusiv, adevărat dacă un operand este adevărat
NOT	NU, adevărat dacă operandul este fals

De exemplu, următoarea interogare raportează rândurile care au un număr de angajat mai mare decât unitatea sau al căror nume include sub-șirul 'George':

```
SELECT angajatnr, nume FROM angajat
WHERE angajatnr > 1 OR nume LIKE '%George%';
```



### Sfatul specialistului

**Întrebare:** Operatorul SQL pentru identificarea echivalenței cu un model nu folosește sintaxa obișnuită a expresiilor regulate.

Există vreun mod de a folosi în SQL expresii regulate?

**Răspuns:** Deși SQL este un limbaj standardizat, distribuitorii SGBD tind să devieze ușor de la limbajul SQL sau să-l extindă. Astfel, fiecare SGBD acceptă un dialect SQL ușor diferit de celelalte. Dialectul acceptat de MySQL include un operator relațional care execută comparația cu o expresie regulată, dar nu toate sistemele de gestiune a bazelor de date includ un asemenea operator.

Pentru a compara o valoare cu o expresie regulată în MySQL, folosiți următoarea formă:

x REGEXP y

unde x este valoarea care va fi testată, iar y este o expresie regulată, delimitată prin ghilimele simple.

MySQL include multe alte extensii ale limbajului SQL. De exemplu, MySQL acceptă următorii operatori ca echivalent:

Operator	Echivalent
AND	&&
OR	
NOT	!

Pentru a vedea și alte diferențe față de standardul SQL și extinderi ale acestuia, consultați manualul MySQL pe suport electronic, la adresa [www.mysql.com](http://www.mysql.com).



### Test „la minut”

- Care este comanda SQL folosită pentru a raporta datele dintr-o bază de date?
- Menționați clauza care vă permite să specificați rândurile raportate de o interogare.
- Precizați operatorul care vă permite să comparați o valoare șir cu un model.

## Modificarea datelor dintr-o bază de date

Sub-limbajul SQL Data Manipulation Language include comenzi care vă permit să inserați rânduri noi într-un tabel, să actualizați una sau mai multe coloane ale rândurilor existente în tabele, respectiv să ștergeți rânduri dintr-un tabel. Pentru a insera un nou rând într-un tabel, folosiți comanda INSERT, care are următoarea formă:

```
INSERT INTO tabel VALUES (valoare1, valoare2);
```

unde tabel este numele tabelului la care se va adăuga rândul, valoare1 este valoarea pentru prima coloană din tabel, iar valoare2 este valoarea celei de-a doua coloane din tabel. Se pot da mai mult sau mai puțin de două valori; numărul valorilor date trebuie să fie egal cu acela al coloanelor din tabel. O coloană poate primi și valoarea NULL, cu excepția situațiilor când definiția coloanei conține specificații contrare.

Răspunsuri la test:

- SELECT
- WHERE
- LIKE

O formă mai populară a comenzii INSERT specifică numele coloanelor cărora le sunt atribuite valorile:

```
INSERT INTO tabel (coloana1, coloana2) VALUES (valoare1, valoare2);
```

În această formă, coloana denumită *coloana1* primește valoarea *valoare1*, iar coloana denumită *coloana2* primește valoarea *valoare2*. Ca în cazul primei forme a comenzii INSERT, pot fi specificate mai mult, respectiv mai puțin de două coloane și valori. Numărul coloanelor specificate trebuie să fie egal cu numărul valorilor specificate. Coloanele care nu sunt denumite în comanda INSERT și care nu au o valoare prestabilită (DEFAULT) primesc valoarea NULL, cu excepția situațiilor când valoarea respectivă nu este permisă; în acest caz, comanda INSERT eșuează.

De exemplu, se poate folosi o comandă similară cu următoarea pentru a insera un rând nou în tabelul angajat:

```
INSERT INTO angajat (angajatnr, nume)
VALUES (4, 'James Monroe');
```

Toate coloanele, cu excepția coloanelor angajatnr și nume, vor primi valoarea NULL.

### Sugestie

Trebuie să evitați utilizarea primei forme a comenzii INSERT. Adăugarea sau ștergerea coloanelor dintr-un tabel pot duce la o funcționare defectuoasă a acestei forme a comenzii, deoarece modul său de operare depinde de echivalența secvențială între valori și coloanele din tabel.

Pentru a modifica valoarea unui rând sau mai multor rânduri existente într-un tabel, emiteți o comandă UPDATE, care are următoarea formă:

```
UPDATE tabel SET coloana1=valoare1, coloana2=valoare2
WHERE conditie;
```

unde *tabel* este numele tabelului ale cărui rânduri urmează să se modifice, *coloana1* este numele primei coloane care urmează a fi modificată, *valoare1* este valoarea care va fi repartizată în *coloana1*, *coloana2* este numele celei de-a doua coloane care urmează a fi modificată, *valoare2* este valoarea care va fi repartizată în *coloana2*, iar *conditie* identifică rândul sau rândurile care urmează a fi actualizate. Poate fi actualizat un număr mai mare sau mai mic de coloane. Dacă urmează ca fiecare rând să fie actualizat, clauza WHERE poate fi omisă.

De exemplu, comanda următoare modifică numele asociat angajatului al cărui atribut angajatnr are valoarea 4 în James Monroe:

```
UPDATE angajat SET nume='James Monroe'
WHERE angajatnr=4;
```

Următoarea comandă mărește salariul fiecărui angajat cu 10 procente:

```
UPDATE angajat SET salariu=1.1*salariu;
```

Pentru a șterge un rând dintr-un tabel, emiteți comanda DELETE, care are următoarea formă:

```
DELETE FROM tabel WHERE conditie;
```

Dacă vor fi șterse toate rândurile tabelului, clauza WHERE poate fi omisă. De exemplu, comanda următoare șterge rândul din tabel asociat angajatului al cărui atribut angajatnr are valoarea 4:

```
DELETE FROM angajat WHERE angajatnr=4;
```

De asemenea, următoarea comandă șterge fiecare rând al tabelului angajat:

```
DELETE FROM angajat;
```

### Sugestie

Comenzile INSERT, UPDATE și DELETE modifică valorile rândurilor din tabel. În general, nu este posibilă recuperarea valorilor originale ale rândurilor din tabel după emiterea uneia dintre aceste comenzi. Ca atare, este important să realizați copii de siguranță ale bazelor de date și să procedați cu atenție la emiterea unor comenzi ca acestea.



### Sfatul specialistului

**Întrebare:** Există vreo modalitate simplă de adăugare a mai multor rânduri într-un tabel dintr-o bază de date?

**Răspuns:** Da. Puteți folosi următoarea formă modificată a comenzii INSERT, care vă permite să specificați mai multe rânduri ale unui tabel:

```
INSERT INTO tabel (coloana1, coloana2)
VALUES
(valoare1, valoare2),
(valoare3, valoare4),
(valoare5, valoare6);
```

Această formă a comenzii vă permite să specificați grupuri de valori; fiecare grup este inclus între paranteze și separat de grupurile adiacente prin intermediul unei virgule. Ca și în cazul formei obișnuite a comenzii INSERT, numărul de coloane specificate trebuie să corespundă cu acela al valorilor specificate în fiecare grup; cu toate acestea, puteți specifica un număr nelimitat de grupuri.



### Test „la minut”

- Scrieți o comandă SQL care înserează un rând nou în tabelul angajat. În noul rând, atributul angajatnr trebuie să aibă valoarea 5, iar atributul nume trebuie să aibă valoarea „James Monroe”.
- Scrieți o comandă SQL care mărește salariul fiecărui angajat cu 20 de procente.

- Scrieți o comandă SQL care șterge rândul din tabel în care atributul angajatnr are valoarea 2.

## Sortarea, agregarea și gruparea

Deseori, este important ca datele să fie raportate într-o anumită secvență. Puteți specifica ordinea de raportare a rezultatelor interogării folosind clauza ORDER BY, care are următoarea formă:

```
ORDER BY valoare
```

Dacă se vor folosi mai multe câmpuri de sortare, fiecare câmp va fi separat de vecinii săi prin intermediul unei virgule. Dacă doriți să indicați o sortare descendentă, în locul uneia ascendente, specificați DESC după valoare. De exemplu, pentru a ordona pe toți angajații în funcție de salariu, de la cel mai mare la cel mai mic, respectiv după nume pentru un salariu dat, puteți folosi următoarea interogare:

```
SELECT salariu, nume FROM angajat
ORDER BY salariu DESC, nume;
```

Pentru a include numai pe angajații care au un salariu mai mare de 50000 USD, adăugați o clauză WHERE la comanda SELECT:

```
SELECT salariu, nume FROM angajat
WHERE salariu>50000
ORDER BY salariu DESC, nume;
```

SQL include funcții care vă permit să raportați valori agregate, precum un număr al rândurilor tabelului. Iată cele mai importante funcții de agregare:

Funcție	Descriere
count(*)	Numărul rândurilor din tabel.
count(coloana)	Numărul rândurilor din tabel care conțin o valoare diferită de NULL în coloana specificată.
count(distinct coloana)	Numărul valorilor distincte diferite de NULL care apar în coloana specificată.
avg(coloana)	Valoarea mijlocie (medie) a coloanei numerice specificate.
min(coloana)	Valoarea minimă din coloana specificată.
max(coloana)	Valoarea maximă din coloana specificată.
sum(coloana)	Suma valorilor din coloana specificată.

De exemplu, interogarea următoare raportează numărul angajaților și salariul mediu al acestora:

```
SELECT count(*), avg(salariu) FROM angajat;
```

Răspunsuri la test:

- INSERT INTO angajat (angajatnr, nume) VALUES (5, 'James Monroe');
- UPDATE angajat SET salariu=1.2\*salariu;
- DELETE FROM angajat WHERE angajatnr=2;

Datele de ieșire vor conține o singură linie, deoarece datele au fost comasate.

Clauza SQL AS specifică un nume nou pentru o coloană sau expresie. Numele specificat este folosit ca titlu în rapoartele SQL. Clauza AS este utilă în lucrul cu funcțiile de agregare. De exemplu, interogarea următoare poate fi rescrisă pentru a include o clauză AS, după cum urmează:

```
SELECT count(*) AS Angajat_Numar,
 avg(salariu) AS Salariu_Mediu
FROM angajat;
```

Rezultatul unei asemenea interogări se poate prezenta astfel:

```
+-----+-----+
| Angajat_Numar | Salariu_Mediu |
+-----+-----+
| 2 | 63250.000000 |
+-----+-----+
1 row in set (0.00 sec)
```

Să presupunem că doriți să afișați numărul angajaților din fiecare departament. Rezultatul unei asemenea interogări va conține o linie pentru fiecare departament, în loc de o linie pentru fiecare angajat. Clauza GROUP BY specifică o asemenea interogare. Clauza are următoarea formă:

```
GROUP BY coloana-sortare
```

unde *coloana-sortare* este numele sau valoarea unei coloane, specificată într-o clauză ORDER BY, care trebuie să urmeze după clauza GROUP BY.

De exemplu, interogarea următoare raportează numărul angajaților și salariul mediu pentru fiecare departament în parte:

```
SELECT count(*), avg(salariu) FROM angajat
GROUP BY deptnr
ORDER BY deptnr;
```

Pentru a include în datele de ieșire numai grupurile selectate, specificați clauza HAVING imediat după clauza GROUP BY. Clauza HAVING are următoarea formă:

```
HAVING conditie
```

De exemplu, pentru a include numai departamentele al căror atribut deptnr are o valoare mai mare decât 2, emiteți următoarea interogare:

```
SELECT count(*), avg(salariu) FROM angajat
GROUP BY deptnr
HAVING deptno>2
ORDER BY deptnr;
```



### Test „la minut”

- Scrieți o interogare care afișează pe toți angajații, ordonați după nume.
- Scrieți o interogare care raportează salariul maxim al angajaților.
- Scrieți o interogare care raportează numărul angajaților din fiecare departament, pentru departamente în care lucrează 10 sau mai mulți angajați.



### Sfatul specialistului

**Întrebare:** Dacă doresc să efectuez sortarea în funcție de o valoare calculată, nu în raport cu valoarea dintr-o coloană? Este posibil acest lucru?

**Răspuns:** Da. Valorile din clauzele ORDER BY și GROUP, precum și condițiile din clauzele WHERE și HAVING pot include și expresii, nu numai simple nume de coloane. În secțiunea următoare vom explica modul de formare a expresiilor SQL pe care le puteți folosi în asemenea contexte.

**Tabelul 13-2** Operatori matematici

Operator	Descriere
+	Adunare
-	Scădere
*	Înmulțire
/	Împărțire
	SAU la nivel de bit
&	ȘI la nivel de bit
>>	Deplasare la dreapta
<<	Deplasare la stânga
~	Complement la nivel de bit

## Expresii și funcții

SQL vă permite să formați expresii folosind valori din coloane, valori literale și funcții. Ca și în PHP, puteți controla ordinea de evaluare a expresiilor SQL folosind paranteze pentru a delimita sub-expresii care trebuie evaluate în prealabil.

Tabelele 13-2 până la 13-6 rezumă operatorii MySQL și funcțiile MySQL frecvent folosite. MySQL furnizează multe alte funcții. Funcțiile incluse au fost selec-

Răspunsuri la test:

- `SELECT nume FROM angajat ORDER BY nume;`
- `SELECT max(salariu) FROM angajat;`
- `SELECT count(*) FROM angajat  
GROUP BY deptnr HAVING count(*)>=10  
ORDER BY deptnr;`

tate în funcție de importanță și de disponibilitatea lor în alte sisteme de gestiune a bazelor de date relaționale. Pentru mai multe informații despre acestea și despre alte funcții MySQL, consultați manualul MySQL pe suport electronic, la adresa <http://www.mysql.com>. Dacă folosiți un alt SGBD decât MySQL, consultați documentația aferentă, pentru a determina funcțiile pe care le acceptă sistemul respectiv.

**Tabelul 13-3** Operatori logici

Operator	Descriere
NOT	NU logic
!	NU logic
OR	SAU logic
	SAU logic
AND	ȘI logic
&&	ȘI logic

**Tabelul 13-4** Funcții matematice frecvent folosite în MySQL

Funcție	Descriere
abs(x)	Valoarea absolută a lui x
atan(x)	Arc tangenta lui x, unde x este dat în radiani
atan2(y, x)	Arc tangenta lui y/x, unde semnele ambelor argumente sunt folosite pentru a determina cadranul cercului trigonometric
ceiling(x)	Cel mai mic întreg care nu este mai mic decât x
cos(x)	Cosinusul lui x, unde x este exprimat în radiani
exp(x)	Baza logaritmilor naturali (e) ridicată la puterea x
floor(x)	Cel mai mare întreg care nu este mai mare decât x
log(x)	Logaritmul natural al lui x
mod(x, y)	Restul împărțirii x/y
power(x, y)	$x^y$
rand(x)	Valoare aleatoare cu virgulă mobilă, mai mare sau egală cu zero și mai mică decât unu
sign(x)	Valoarea -1, 0 sau 1, după cum valoarea lui x este negativă, zero sau pozitivă
sin(x)	Sinusul lui x, unde x este dat în radiani
sqrt(x)	Rădăcina pătrată a lui x
tan(x)	Tangenta lui x, unde x este dat în radiani

**Tabelul 13-5** Funcții șir frecvent folosite în MySQL

Funcții	Descriere
ascii(s)	Codul ASCII al octetului celui mai din stânga al șirului s
char(n)	Caracter al cărui cod ASCII este n
concat(s1, s2)	Concatenarea șirurilor s1 și s2; cu alte cuvinte, s2 atașat la s1
lcase(s)	Șirul s, unde toate majusculele au fost transformate în minuscule
left(s, n)	Primii n octeți ai șirului s, de la stânga la dreapta
length(s)	Numărul octeților din șirul s



Tabelul 13-5: Continuare

Funcții	Descriere
locate(s1, s2)	Poziția primei apariții a lui s1 în s2, respectiv zero dacă s1 nu se găsește în s2
ltrim(s)	Șirul s, cu eliminarea spațiilor de început
right(s, n)	Primii n octeți din șirul s, de la dreapta la stânga
rpad(s1, n, s2)	Șirul s1, completat la dreapta cu șirul s2 până când rezultatul are lungimea n
rtrim(s)	Șirul s, cu spațiile finale eliminate
space(n)	Un șir alcătuit din n spații
substring(s, m, n)	Sub-șir al lui s, care începe de la poziția m și care are lungimea n
trim	Sub-șir al lui s, cu spațiile inițiale și finale eliminate
ucase(s)	Șirul s, cu toate minusculele convertite în majuscule

Tabelul 13-6: Funcții MySQL de dată și oră frecvent utilizate

Funcție	Descriere
dayofmonth(d)	Ziua din lună a datei specificate (1-31)
dayofweek(d)	Ziua din săptămână a datei specificate (1=duminică, 2=luni, ..., 7=sâmbătă)
dayofyear(d)	Ziua din an a datei specificate (1-366)
hour(t)	Partea orelor din momentul de timp menționat (0-23)
minute(t)	Partea minutelor din momentul de timp menționat (0-59)
month(d)	Luna datei specificate (1-12)
now()	Data și ora curentă
second(t)	Partea secundelor din momentul de timp menționat (0-59)
week(d)	Săptămâna din an a datei specificate (0-53)
year(d)	Partea anilor din momentul de timp menționat (1000-9999)

## ? Sfatul specialistului

**Întrebare:** Care sunt unele din funcțiile acceptate de MySQL, dar care nu sunt acceptate pe scară largă de alte sisteme de gestiune a bazelor de date?

**Răspuns:** Ca și ale sisteme de gestiune a bazelor de date, MySQL furnizează o suită de funcții care execută operații inexistente în standardul SQL. Asemenea funcții sunt complet diferite de la un SGBD la altul. Iată câteva dintre principalele funcții furnizate de MySQL care nu fac parte din standardul SQL:

Funcție	Descriere
database()	Returnează numele bazei de date deschise
get_lock(s, n)	Obține o blocare a bazei de date
md5(s)	Returnează o sumă de control a șirului s, calculată după algoritmul MD5
password(s)	Returnează șirul s, criptat folosind algoritmul aplicat de MySQL parolei
release_lock(s)	Anulează blocarea unei baze de date
user()	Returnează numele utilizatorului curent
version()	Returnează numărul versiunii MySQL

Funcțiile `get_lock()` și `release_lock()` vă permit să controlați accesul la tabelele dintr-o bază de date. Aceste funcții sunt importante deoarece MySQL nu implementează integritatea tranzațiilor, lipsă care poate duce la pierderea unor actualizări sau la deteriorarea datelor atunci când utilizatorii obțin accesul într-o manieră concurențială la date corelate. Totuși, utilizarea acestor funcții este un subiect complex, care depășește cu mult „aria de acoperire” a volumului de față. Pentru alte informații, consultați orice manual detaliat de teoria bazelor de date sau examinați manualul MySQL pe suport electronic, la adresa [www.mysql.com](http://www.mysql.com).



### Test „la minut”

- Care este funcția MySQL ce returnează o versiune a unui șir scrisă cu minuscule?
- Care este funcția MySQL care elimină dintr-un șir spațiile inițiale și finale?
- Care este funcția MySQL ce returnează data și ora curentă?

## Uniri

SQL vă permite să obțineți accesul la mai multe tabele într-o singură interogare. În general, această operație se execută pentru a urma relația stabilită printr-o cheie externă, făcând ca datele din tabelul corelat să fie disponibile în interogare. De exemplu, să presupunem că baza de date este asemănătoare celei prezentate în figura 13-2, unde o relație cheie externă – cheie primară asociază tabelele angajat și meserii. Să examinăm următoarea interogare:

```
SELECT nume, meserie FROM angajat, meserii
WHERE angajat.angajatnr = meserii.angajatnr;
```

Construcțiile `angajat.angajatnr` și `meserii.angajatnr` se numesc *nume definite*, prima se referă la coloana `angajatnr` a tabelului `angajat`, iar a doua se referă la coloana `angajatnr` a tabelului `meserii`. Clauza `WHERE` asigură o echivalență adecvată între valoarea cheii externe din tabelul `meserii` cu aceea a cheii primare din tabelul `angajat`. În absența clauzei `WHERE`, se va stabili o corespondență între fiecare rând din tabelul cu `meserii` și fiecare rând din tabelul cu `angajați`. Un asemenea rezultat, numit produs cartezian, conține în general multe rânduri – majoritatea nedorite – și ca atare trebuie evitat.

Rezultatul acestei interogări este un raport care indică numele și meseria asociată fiecărui angajat prezentat în tabelul `meserii`:

Răspunsuri la test:

- `lcase()`
- `trim()`
- `now()`

nume	meserie
George Washington	General
John Adams	Filosof
T. Jefferson	Arhitect

3 rows in set (0.00 sec)

O interogare ca aceasta, care combină date provenite din mai multe tabele, se numește *unire*. Sunt posibile și uniri mai complexe, care implică trei sau mai multe tabele.

Într-o unire din două tabele, unii denumesc tabelul care conține cheia primară ca *tabel master*, iar tabelul care conține cheia externă ca *tabel cu detalii*. O asemenea relație între tabele este numită *relație master-detalii*. Într-o asemenea relație, un singur rând din tabelul master poate fi asociat cu mai multe rânduri din tabelul cu detalii.

## ? Sfatul specialistului

**Întrebare:** Dacă o relație cheie externă – cheie primară este opțională, adică are cardinalitatea 0:1? Dacă un rând dat din tabelul master nu are nici un rând asociat în tabelul cu detalii, rândul respectiv din tabelul master nu va apărea în datele de ieșire ale unei uniri. Există vreo metodă de a rezolva această problemă și de a determina apariția înregistrării din tabelul master?

**Răspuns:** Da. Puteți folosi o categorie specială de unire, cunoscută sub numele de *unire la stânga* sau *unire exterioară la stânga*. Iată un exemplu:

```
SELECT nume, meserie FROM angajat
LEFT JOIN meserii
ON angajat.angajatnr=meserii.angajatnr;
```

Efectul acestei interogări constă în afișarea tuturor angajaților, indiferent dacă la aceștia este sau nu asociată o meserie. Angajații fără o meserie au specificația NULL în coloana corespunzătoare meseriei:

nume	meserie
George Washington	General
John Adams	NULL
T. Jefferson	Arhitect

3 rows in set (0.00 sec)

Mulți oameni – poate majoritatea – sunt de părere, la început, că unirile la stânga sunt derutante, deci nu vă supărați dacă ați intrat în încurcătură. Pentru mai multe informații referitoare la unirile la stânga, examinați un manual detaliat de teoria bazelor de date.



## Test „la minut”

- Să considerăm o bază de date care include un tabel numit *carte*, a cărui cheie primară este ISBN, precum și un tabel numit *vanzare*, a cărui cheie primară este *idtranzactie* și a cărui cheie externă este ISBN. Scrieți o interogare care unește cele două tabele.

## Proiect 13-1: Lucrul cu limbajul SQL

În cadrul acestui proiect, veți învăța să creați și să manipulați o bază de date MySQL. Veți construi o bază de date demonstrativă, pe care o puteți folosi pentru a exersa alcătuirea interogărilor SQL.

### Scopurile proiectului

- Explicarea modului de creare a unei baze de date MySQL
- Prezentarea modului de utilizare a programului `mysql` pentru a emite interogări SQL interactive
- Furnizarea unei baze de date demonstrative pentru a învăța mai multe despre interogările SQL

### Pas cu pas

1. Dacă folosiți un furnizor de servicii Internet care a creat deja o bază de date MySQL pentru uzul dumneavoastră, atunci treceți la etapa 4. În caz contrar, deschideți sesiunea de lucru ca administrator de sistem sau cereți administratorului de sistem să execute următoarea operație pentru dumneavoastră.
2. Creați următorul script de shell, denumindu-l `p13-1a.sh`:

```
mysql -p <<EOF
CREATE DATABASE testdb;
USE testdb;
GRANT ALL ON testdb.* TO php IDENTIFIED AS 'salut';
EOF
```

Acest script creează o bază de date și un utilizator care are privilegii complete pentru accesul la baza de date și manipularea acesteia. Dacă doriți, puteți înlocui numele bazei de date (`testdb`), numele utilizatorului (`php`) sau parola (`salut`) cu valorile pe care le preferați. Puteți specifica numele dumneavoastră de utilizator și parola.

3. Executați scriptul, prin emiterea comenzii:

```
sh p13-1a.sh
```

Comanda `mysql` vă va solicita parola administratorului de sistem. Introduceți valoarea adecvată și apăsați pe tasta ENTER. Când scriptul și-a încheiat execuția, veți primi un prompt de shell.

4. Creați următorul script de shell, denumindu-l `p13-1b.sh`:

```
mysql -p -u php <<EOF
USE testdb
```

Răspuns la test:

- `SELECT * FROM carte, vanzare`  
`WHERE carte.ISBN=vanzare.ISBN`

```

CREATE TABLE angajat
(
 angajatnr SMALLINT PRIMARY KEY,
 nume VARCHAR(50),
 ore SMALLINT,
 departament CHAR(16),
 salariu DECIMAL(8,2),
 data_angajare DATE
);

INSERT INTO angajat
(
 angajatnr,
 nume,
 ore,
 departament,
 salariu,
 data_angajare
)
VALUES
(
 1,
 'George Washington',
 40,
 'Contabilitate',
 80000.00,
 '2005-10-01'
),
(
 2,
 'John Adams',
 35,
 'Marketing',
 20000.00,
 '2005-10-15'
),
(
 3,
 'Thomas Jefferson',
 20,
 'Vanzari',
 35000.00,
 '2005-09-01'
);

CREATE TABLE meserii
(
 meserie CHAR(16), PRIMARY KEY,
 angajatnr SMALLINT
);

INSERT INTO meserii
(
 meserie,
 angajatnr
)
VALUES

```

```

(
 'General',
 1
),
(
 'Filosof',
 2
),
(
 'Arhitect',
 3
);
EOF

```

Acest script creează două tabele în baza de date și inserează mai multe rânduri ale unui tabel. Fiți foarte atent cum introduceți textul scriptului de la tastatură. Mai bine descărcați scriptul din site-ul Web al acestei cărți, pentru a evita problemele. Dacă ați modificat baza de date sau numele utilizatorului în etapa 2, efectuați aici schimbările corespunzătoare.

5. Executați scriptul prin emiterea comenzii:

```
sh p13-1b.sh
```

Programul mysql vă va solicita parola pe care ați introdus-o în scriptul construit în etapa 2 (salut). Introduceți parola și apăsați pe tasta ENTER. Când execuția scriptului s-a încheiat, veți primi un prompt de shell.

6. Acum, sunteți pregătit să emiteți interogări interactive asupra bazei de date. Pentru aceasta, emiteți comanda:

```
mysql -p -u php
```

Când vi se cere, răspundeți cu parola pe care ați specificat-o în etapa 2 (salut).

7. Emiteți câteva interogări, cum sunt următoarele, asupra bazei de date:

```

SELECT * FROM angajat;
SELECT * FROM meserii;

```

8. Acum, încercați unele interogări mai ambițioase, create de dumneavoastră. Interfața cu utilizatorul este asemănătoare cu linia de comandă Linux; puteți folosi tastele cu săgeți pentru a reexecuta și edita comenzile emise anterior.

9. Iată unele interogări non-SQL posibil utile pe care le puteți încerca:

```

SHOW STATUS;
SHOW DATABASES;
SHOW TABLES;
SHOW COLUMNS FROM testdb;
SHOW INDEX FROM testdb;
SHOW CREATE TABLE testdb;
SHOW GRANTS FOR php;

```

Aceste interogări, care sunt specifice sistemului MySQL, prezintă starea și structura bazelor de date și a tabelor. Multe din aceste interogări produc un volum mare de date de ieșire. Pentru informații referitoare la interpretarea datelor de ieșire, consultați manualul SQL pe suport electronic, de la adresa [www.mysql.com](http://www.mysql.com).

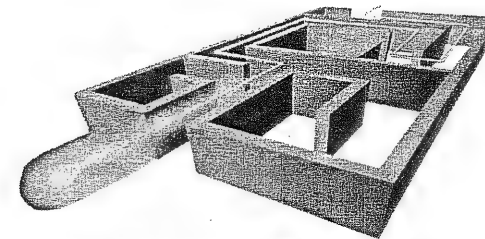
10. Când ați terminat cu interogările, părăsiți programul mysql prin emiterea comenzii:

quit



### Test de evaluare

1. Cum se numește componenta unei baze de date relaționale care conține date referitoare la o instanță a unei entități?
2. Cum se numește tipul de cheie care nu este, în general, unică pentru fiecare rând al unui tabel dintr-o bază de date?
3. Care este cardinalitatea tipului de relație care trebuie eliminată în cursul procesului de modelare E-R?
4. Scrieți o comandă SQL care creează un tabel denumit test, care conține două câmpuri de câte 16 caractere fiecare, numite a și b.
5. Scrieți o comandă SQL care inserează în baza de date creată la întrebarea anterioară un rând având ca valoare un șir de spații.
6. Scrieți o comandă SQL care raportează toate rândurile incluse în baza de date creată la întrebarea nr. 4.



## Partea a IV-a:

# Utilizarea funcționalităților avansate ale limbajului PHP

## Modulul 14:

# Accesul la bazele de date relaționale

### Scopuri

- Învățați să vă conectați la un server de baze de date MySQL
- Învățați să executați interogări SQL asupra unei baze de date MySQL
- Învățați să detectați și să raportați erori în baza de date
- Învățați să obțineți informații privind rezultatele interogărilor SQL
- Învățați să obțineți informații despre structura unei baze de date MySQL
- Învățați să vă protejați aplicațiile împotriva anumitor categorii de date rău intenționate introduse de utilizator
- Înțelegeți caracteristicile inexistente în MySQL furnizate de instrumente de gestiune a datelor, precum Postgresql, ODBC, LDAP și XML

Datele sunt obiectul celor mai multe operații de prelucrare, iar sistemele de gestiune a bazelor de date furnizează cele mai complexe și mai puternice facilități pentru lucrul cu datele. Ca atare, un programator PHP trebuie să dispună de cunoș-

tințe aprofundate privind sistemele de gestiune a bazelor de date. Acest modul explică modul de redactare a programelor PHP care folosesc MySQL, sistemul de gestiune a bazelor de date cel mai frecvent folosit de către programatorii PHP. De asemenea, sunt descrise și alte instrumente de gestiune a datelor, inclusiv Postgresql, ODBC, LDAP și XML.

## Utilizarea bazelor de date MySQL

✦ PHP include o bibliotecă de funcții care furnizează o interfață cu sistemul MySQL de gestiune de baze de date. Folosind aceste funcții, un program PHP poate obține accesul la datele rezidente într-o bază de date MySQL și le poate modifica.

Majoritatea interacțiunilor cu o bază de date se desfășoară după un model secvențial simplu:

1. Se deschide o conexiune cu serverul MySQL.
2. Se specifică baza de date la care se va obține accesul.
3. Se emit interogări SQL, se obține accesul la rezultatele interogărilor și se execută operații non-SQL.
4. Se închide conexiunea cu serverul MySQL.

Această secțiune descrie deschiderea unei conexiuni cu o bază de date, specificarea bazei de date la care urmează a se obține accesul și închiderea conexiunii cu serverul MySQL. De asemenea, se explică modul de detectare a erorilor în procesul de prelucrare a interogărilor MySQL și modalitățile de răspuns în cazul apariției acestora. În secțiunile următoare, se explică modul de emiteră a interogărilor SQL, de acces la rezultatele interogărilor și de execuție a operațiilor non-SQL.

## Conectarea la serverul MySQL

Pentru a vă conecta la un server MySQL, invocați funcția `mysql_connect()`, a cărei sintaxă este următoarea:

```
mysql_connect(nume_gazda, nume_utilizator, parola)
```

unde *nume\_gazda* este numele gazdei pe care rulează serviciul MySQL, *nume\_utilizator* este identificatorul de utilizator MySQL care va fi folosit, iar *parola* este parola MySQL asociată identificatorului de utilizator. Funcția returnează false în caz de eșec; în caz contrar, returnează o valoare – denumită *identificator de legătură* – care servește ca instrument de manipulare pentru accesul la serverul MySQL.

Iată un model de invocare a funcției `mysql_connect()`:

```
$db = mysql_connect("localhost", "php", "salut");
if (!$db)
 die("Nu s-a reusit deschiderea bazei de date.");
```

Exemplul anterior testează valoarea rezultatului returnat de funcția `mysql_connect()` și încheie execuția programului dacă PHP nu a reușit să deschidă conexiunea specificată. Argumentele prezentate în exemplu sunt adecvate pentru conectarea la un server MySQL care rulează pe aceeași gazdă ca și serverul PHP, adică gazda locală. Identificatorul de utilizator și parola sunt similare celor specificate în proiectul 13-1 din modulul anterior.

Puteți omite numele gazdei, identificatorul de utilizator și parola, sau toate cele trei argumente. Dacă procedați astfel, vor fi luate în considerare în mod prestabilit următoarele valori:

- Numele gazdei: localhost
- Identificatorul de utilizator: identificatorul de utilizator al procesului server MySQL.
- Parolă: o parolă vidă

De exemplu, instrucțiunea următoare încearcă să stabilească o conexiune cu serviciul MySQL care rulează pe gazda `db.osborne.com`, folosind un identificator de utilizator și o parolă prestabilite:

```
$db = mysql_connect("db.osborne.com");
```

## Sugestie

În mod prestabilit, funcția `mysql_connect()` încearcă să contacteze serviciul MySQL prin intermediul portului 3306, portul MySQL standard. Dacă doriți să obțineți accesul la un server MySQL care rulează pe un port non-standard, puteți atașa un caracter două puncte și numărul portului dorit la argumentul care conține numele gazdei; de exemplu, `"localhost:3305"`.

## Selectarea bazei de date

După ce programul dumneavoastră a obținut o conexiune cu serverul MySQL, programul poate specifica baza de date la care va avea acces. Pentru aceasta, invocă funcția `mysql_select_db()`, care are următoarea formă:

```
mysql_select_db(baza_de_date)
```

unde *baza\_de\_date* este un șir care conține numele bazei de date la care urmează a se obține acces. Funcția returnează true dacă poate obține accesul la baza de date, respectiv false în caz contrar.

Puteți testa rezultatul funcției `mysql_select_db()` folosind un program ca acesta:

```
$ok = mysql_select_db("testdb");
if (!$ok)
{
 die("Nu poate obtine acces la baza de date testdb.");
}
```

Totuși, acest procedeu nu este foarte util pentru a determina cauza sau natura unei invocări ratate. O metodă mai bună constă în utilizarea funcțiilor din biblioteca MySQL de verificare a erorilor, funcții descrise în secțiunea următoare, „Detectarea apariției erorilor“.

## Detectarea apariției erorilor

Biblioteca MySQL din PHP furnizează două funcții de verificare a erorilor, și anume `mysql_errno()` și `mysql_error()`. Fiecare funcție returnează un rezultat care reflectă eroarea; dacă există, asociată celei mai recente operații cu MySQL. Dacă programul dumneavoastră execută o secvență de operații MySQL, iar prima operație generează o eroare, informațiile despre erorile respective sunt pierdute în momentul inițierii celei de-a doua operații.

Nici una din cele două funcții nu necesită argumente. Funcția `mysql_errno()` returnează un cod numeric de eroare, în timp ce funcția `mysql_error()` returnează o descriere textuală a erorii. Dacă nu s-a produs nici o eroare, codul numeric al erorii este zero și descrierea textuală are ca valoare un șir vid.

Informațiile de eroare sunt disponibile numai dacă este activă o conexiune cu serverul MySQL. Ca atare, nu puteți folosi nici una dintre aceste funcții pentru a raporta erorile asociate funcției `mysql_connect()`.

Iată cum puteți folosi funcțiile respective pentru a verifica modul de operare a funcției `mysql_select_db()`:

```
mysql_select_db("testdb");
if(mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
```

De exemplu, dacă încercați să obțineți accesul la baza de date inexistentă `testdbx`, programul de mai sus va genera următorul rezultat:

```
1044: Access denied for user: 'php@localhost' to database 'testdbx'
(Accesul interzis pentru utilizatorul ... la baza de date ...)
```

## Eliminarea mesajelor de eroare și a avertismentelor nedorite

Numeroase funcții PHP pot produce erori sau mesaje de avertizare care îi pot deruta pe utilizatorii siturilor Web sau care le pot cauza neplăceri acestora. PHP furnizează funcția `error_reporting()`, care vă permite să eliminați mesaje nedorite. Funcția are următoarea formă:

```
error_reporting(masca);
```

unde *masca* specifică tipul mesajelor care vor fi raportate. Dacă specificați zero ca valoare a atributului *masca*, nu va fi raportat nici un mesaj. Dacă specificați `E_ALL` ca

valoare a atributului *masca*, vor fi raportate toate mesajele. De exemplu, pentru a elimina toate mesajele, invocați funcția după cum urmează:

```
error_reporting(0);
```

În general, este util să permiteți limbajului PHP să afișeze mesaje de eroare și de avertisment în faza de dezvoltare a programelor, deoarece acestea vă pot ajuta să identificați și să eliminați problemele. Ca atare, în general trebuie să eliminați erorile și mesajele de avertisment numai pentru programele aflate în uz, nu și pentru cele aflate în faza de dezvoltare.

## Închiderea conexiunii cu serverul MySQL

Pentru a închide o conexiune cu un server MySQL, invocați funcția `mysql_close()`, care are următoarea formă:

```
mysql_close();
```

Funcția returnează `true` în caz de reușită; în caz contrar, returnează `false`. În general, nu este necesară invocarea funcției `mysql_close()`, deoarece PHP închide automat conexiunile deschise cu bazele de date atunci când un script își încheie execuția.

Iată cum se poate folosi funcția `mysql_close()` pentru a închide o conexiune:

```
mysql_close();
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
```



### Sfatul specialistului

**Întrebare:** Dacă PHP închide în mod automat conexiunile deschise cu baza de date atunci când un script își încheie execuția, de ce este necesară invocarea funcției `mysql_close()`?

**Răspuns:** Prin închiderea unui fișier, sunt eliberate resursele alocate. Închiderea unei conexiuni cu o bază de date eliberează de asemenea resursele alocate. Dacă închideți o conexiune cu o bază de date înainte de sfârșitul programului dumneavoastră, resursele suplimentare se vor găsi la dispoziția altor procese. Ocazional, puteți scrie un program care obține accesul la mai multe servere MySQL. Într-o asemenea situație, puteți păstra o singură conexiune deschisă la orice moment de timp dat; funcția `mysql_close()` vă permite să închideți o conexiune cu o bază de date anterior deschiderii unei alte conexiuni.





### Test „la minut”

- Care este funcția utilizată pentru a deschide o conexiune cu o bază de date MySQL?
- Care este funcția utilizată pentru a specifica baza de date MySQL la care se va obține accesul folosind o anumită conexiune cu o bază de date?
- Care este funcția ce returnează codul numeric de eroare asociat celei mai recente operațiuni MySQL?

## Executarea interogărilor UPDATE, INSERT și DELETE

Din punctul de vedere al limbajului PHP, există două categorii de interogări SQL:

- Interogările SELECT, care returnează rânduri ale unui tabel
- Interogările UPDATE, INSERT și DELETE, care nu returnează rânduri ale unui tabel

Ambele categorii de interogări sunt emise folosind funcția `mysql_query()`, dar verificarea și prelucrarea celor două categorii de rezultate ale interogărilor sunt procese destul de diferite. În secțiunea următoare este explicat modul de utilizare a funcției `mysql_query()`. De asemenea, este explicat modul de utilizare a interogărilor care nu returnează rânduri de tabel. Dacă vă interesează rezultatul invers, examinați o secțiune ulterioară din acest modul, intitulată „Prelucrarea rezultatelor interogărilor SELECT”, în care se explică modul de prelucrare a rezultatelor interogărilor care returnează rânduri de tabel.

## Funcția `mysql_query()`

Funcția `mysql_query()` execută o interogare specificată. Funcția are următoarea formă:

```
mysql_query(interogare)
```

unde *interogare* este un șir care conține interogarea care urmează a fi executată (interogarea nu trebuie să se încheie cu un caracter punct și virgulă). Funcția returnează `true` dacă serverul a reușit să execute interogarea; în caz contrar, returnează `false`.

Iată o invocare caracteristică a funcției `mysql_query()`, care include un program ce verifică dacă interogarea a reușit sau nu:

Răspunsuri la test:

- `mysql_connect()`
- `mysql_select_db()`
- `mysql_errno()`

```
$interogare = "INSERT INTO angajat
(angajatnr, nume, ore, departament, salariu, data_angajare)
VALUES (4, 'James Madison', 40, 'Intretinere', 20000, '2005-10-01')";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
```

Interogarea este compatibilă cu structura bazei de date folosită în proiectul 13-1, deci puteți rula atât această interogare, cât și alte interogări similare, pentru a vedea cum funcționează.

## Verificarea interogărilor care nu returnează rânduri de tabel

Pentru a verifica dacă o interogare UPDATE, INSERT sau DELETE a avut efectul dorit, puteți folosi funcția `mysql_affected_rows()`, care returnează numărul rândurilor afectate de interogarea cea mai recentă. Funcția are următoarea formă:

```
mysql_affected_rows()
```

În cazul în care cea mai recentă interogare UPDATE, INSERT sau DELETE a eșuat, funcția returnează valoarea `-1`.

Iată cum puteți folosi funcția `mysql_affected_rows()` pentru a determina modul de funcționare a interogării INSERT date anterior:

```
$interogare = "INSERT INTO angajat
(angajatnr, nume, ore, departament, salariu, data_angajare)
VALUES (4, 'James Madison', 40, 'Intretinere', 20000, '2005-10-01')";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
if (mysql_affected_rows() != 1)
{
 die("
INSERT nu a reusit sa adauge angajatul.");
}
```

## Sugestie

Funcția `mysql_affected_rows()` numără numai rândurile efectiv modificate de către o interogare UPDATE. Rândurile în cazul cărora vechea și noua valoare din coloană sunt identice nu se numără printre rândurile afectate. De asemenea, o interogare DELETE care nu conține o clauză WHERE va determina funcția `mysql_affected_rows()` să returneze valoarea zero, indiferent de numărul rândurilor șterse din tabel.

## Utilizarea coloanelor de tabel cu auto-incrementare

Așa cum s-a arătat în modulul anterior, puteți folosi indicatorul `AUTO_INCREMENT` pentru a preciza faptul că MySQL va repartiza o valoare secvențială unică în coloana care servește drept cheie primară a tabelului. De exemplu, următoarele instrucțiuni SQL creează un tabel cu același tip de coloană:

```
CREATE TABLE master
(
 id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
 nume VARCHAR(50)
);
```

Când inserați un rând într-un tabel în acest mod, puteți folosi funcția `mysql_insert_id()` pentru a determina valoarea cheii primare atribuite de MySQL. Funcția are forma

```
mysql_insert_id()
```

și returnează valoarea zero dacă interogarea precedentă nu a generat o valoare `AUTO_INCREMENT`. Ca atare, funcția trebuie apelată la puțin timp după interogarea care a inserat rândul din tabel, astfel încât o interogare ulterioară să nu modifice rezultatul.

Iată cum se poate insera un rând în tabelul master și cum se poate obține valoarea cheii primare repartizate de MySQL:

```
$interogare = "INSERT INTO master (nume) VALUES ('G.W.Bush')";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno() . ": " . mysql_error() . "
");
}
echo "
Randuri modificate: " . mysql_affected_rows();
echo "
ID inserat: " . mysql_insert_id();
```



### Sfatul specialistului

**Întrebare:** Funcția `mysql_affected_rows()` poate returna zero atunci când sunt șterse toate rândurile asociate unui tabel. Cum se poate afla dacă o operație de acest gen a reușit?

**Răspuns:** O modalitate simplă, dar fiabilă, de a determina dacă ștergerea tuturor rândurilor unui tabel a reușit constă în a emite o interogare care returnează numărul rândurilor existente în tabel. De exemplu:

```
SELECT COUNT (angajatnr) FROM angajat;
```

Dacă interogarea returnează valoarea zero, acest fapt demonstrează ștergerea tuturor rândurilor din tabel.

### Atenție

Funcția `mysql_insert_id()` poate returna un rezultat incorect pentru coloanele de tipul MySQL `BIGINT`. Secțiunea "Sfatul specialistului" de la sfârșitul următoarei secțiuni, „Prelucrarea rezultatului interogărilor SELECT”, descrie un procedeu de rezolvare a problemei.



### Test „la minut”

- Care este funcția PHP folosită pentru a emite o interogare MySQL?
- Care este funcția PHP ce returnează numărul rândurilor unui tabel modificate de o interogare `UPDATE`, `INSERT` sau `DELETE`?
- Care este indicatorul MySQL ce arată că este necesar ca valoarea unei chei primare să fie atribuită de către MySQL?

## Prelucrarea rezultatelor interogărilor SELECT

Spre deosebire de interogările `UPDATE`, `INSERT` și `DELETE`, interogările `SELECT` returnează rânduri de tabel ca rezultate. Rândurile unui tabel sunt incluse într-o structură de date numită *set de rezultate*. Prelucrarea setului de rezultate returnat de o interogare `SELECT` implică parcurgerea prin iterație a rândurilor setului de rezultate.

O modalitate de parcurgere iterativă a rândurilor unui set de rezultate constă în obținerea numărului de rânduri, urmată de deplasarea prin iterație, folosind numărul de rânduri ca limită pentru o instrucțiune `for`. Pentru a obține valoarea numărului de rânduri, invocați funcția `mysql_num_rows()`, transferând ca argument valoarea returnată de funcția `mysql_query()`. De exemplu:

```
$interogare = "SELECT * FROM angajat";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno() . ": " . mysql_error() . "
");
}
$numar = mysql_num_rows($rezultat);
```

Funcția `mysql_fetch_row()` se poate folosi pentru a obține următorul rând din secvența setului de rezultate, astfel:

```
for ($i = 0; $i < $numar; $i++)
{
 $rand = mysql_fetch_row($rezultat);
 if (mysql_errno())
 {
 die("
" . mysql_errno() . ": " . mysql_error() . "
");
 }
}
```

Răspunsuri la test:

- `mysql_query()`
- `mysql_affected_rows()`
- `AUTO_INCREMENT`

```

}
// aici se prelucrează randul din setul de rezultate
}

```

Totuși, funcția `mysql_fetch_row()` returnează `true` dacă un set de rezultate conține rânduri neprelucreate, respectiv `false` în caz contrar. Ca atare, în general este mai convenabil să se omită apelarea funcției `mysql_num_rows()` și să se folosească în schimb o instrucțiune `while`, astfel:

```

$interogare = "SELECT * FROM angajat";
$resultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": " . mysql_error(). "
");
}

while ($rand = mysql_fetch_row($resultat))
{
 if (mysql_errno())
 {
 die("
" . mysql_errno().": " . mysql_error(). "
");
 }
 // aici se prelucrează randul din setul de rezultate
}

```

Valoarea returnată de funcția `mysql_fetch_row()` reprezintă un tablou alcătuit din toate coloanele rândului curent din tabel. Tabloul folosește indexuri întregi, unde valoarea primului index este egală cu zero. Pentru a prelucra coloanele stocate în tablou, folosiți o instrucțiune `foreach`, care elimină necesitatea existenței unui index explicit al buclei. De exemplu, iată o instrucțiune `foreach` care pur și simplu afișează valoarea din fiecare coloană a tabelului:

```

while ($rand = mysql_fetch_row($resultat))
{
 if (mysql_errno())
 {
 die("
" . mysql_errno().": " . mysql_error(). "
");
 }
 foreach ($rand as $coloana)
 {
 echo "
$coloana";
 }
 echo "
";
}

```

Dacă doriți să obțineți acces la valoarea unei anumite coloane, puteți face referire la elementul din tablou folosind un index. De exemplu, dacă rezultatul funcției `mysql_fetch_row()` este stocat în variabila `$rand`, puteți obține acces la prima coloană din setul de rezultate folosind sintaxa `$rand[0]`, la a doua coloană folosind sintaxa `$rand[1]` etc.

Dacă vi se pare incomod să lucrați cu indici numerici, puteți obține rândurile tabelului folosind funcția `mysql_fetch_array()`, care returnează un tablou asociativ. Valorile indexurilor din tablou le reprezintă numele coloanelor din setul de rezul-

tate. Ca și funcția `mysql_fetch_row()`, funcția `mysql_fetch_array()` returnează `false` dacă nu mai există rânduri în setul de rezultate.

Iată un exemplu de utilizare a funcției `mysql_fetch_array()`:

```

$interogare = "SELECT * FROM angajat";
$resultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": " . mysql_error(). "
");
}

while ($rand = mysql_fetch_array($resultat, MYSQL_ASSOC))
{
 if (mysql_errno())
 {
 die("
" . mysql_errno().": " . mysql_error(). "
");
 }
 foreach ($rand as $nume=>$coloana)
 {
 echo "
$nume=>$coloana";
 }
 echo "
";
}

```

La rulare, exemplul afișează numele și valoarea fiecărei coloane din setul de rezultate. Dacă doriți să obțineți accesul la valoarea unei anumite coloane, folosiți numele coloanei ca index. De exemplu:

```

echo "
" . $row["angajatnr"];

```

## Sugestie

Cel de-al doilea argument al funcției `mysql_fetch_array()` este opțional. Totuși, dacă nu specificați `MYSQL_ASSOC` ca valoare a argumentului, PHP returnează un tablou asociativ, indexat cu numerele și numele coloanelor.



## Test „la minut”

- Care este numele structurii de date asociate cu rezultatele unei interogări `SELECT`?
- Care este funcția PHP MySQL ce returnează numărul rândurilor dintr-un set de rezultate?
- Care este funcția PHP MySQL ce returnează un tablou cu indexuri numerice, care conține rândul unui set de rezultate?
- Care este funcția PHP MySQL ce returnează un tablou asociativ, care conține un rând al unui set de rezultate, indexat cu numele coloanelor din setul de rezultate?

Răspunsuri la test:

- Set de rezultate
- `mysql_num_rows()`
- `mysql_fetch_row()`
- `mysql_fetch_array()`



## Sfatul specialistului

**Întrebare:** Anterior în cadrul acestui modul, s-a precizat că funcția `mysql_insert_id()` poate fi uneori nesigură. Cum este posibilă o determinare sigură a valorii atribuite de MySQL drept cheie primară cu auto-incrementare a rândului unui tabel?

**Răspuns:** Funcția MySQL `LAST_INSERT_ID()` returnează valoarea atribuită de MySQL unei coloane `AUTO_INCREMENT`, indiferent de tipul coloanei. Mai mult, apelurile ulterioare, la funcții MySQL nu invalidează rezultatul returnat de `LAST_INSERT_ID()`, care este afectat numai de operațiile `INSERT` în care sunt implicate coloane `AUTO_INCREMENT`.

Iată un exemplu care prezintă modul de obținere a valorii `LAST_INSERT_ID()`:

```
$interogare = "SELECT COUNT(*) LAST_INSERT_ID() FROM numetabel";
$resultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
$row = mysql_fetch_row($resultat);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
echo "
ID repartizat: $row[0]";
```

Pentru a folosi aceste linii în propriul dumneavoastră program, înlocuiți `numetabel` cu numele tabelului actualizat.

## Lucrul cu seturi de rezultate

Biblioteca de funcții MySQL a limbajului PHP include un set de funcții care vă permit să obțineți informații despre un set de rezultate, inclusiv:

- Numărul coloanelor din setul de rezultate
- Numărul fiecărei coloane
- Lungimea fiecărei coloane
- Indicatorii MySQL asociați coloanei
- Tipul MySQL al fiecărei coloane
- Numele tabelului MySQL care conține coloana, dacă este cazul

De asemenea, biblioteca furnizează o funcție care vă permite să obțineți accesul în mod non-secvențial la rândurile din setul de rezultate, prin specificarea numărului unui rând.

## Obținerea numărului coloanelor unui set de rezultate

Pentru a obține numărul coloanelor dintr-un set de rezultate, invocați funcția `mysql_num_fields()`, transferând ca argument valoarea returnată de funcția `mysql_query()`.

De exemplu, programul următor folosește funcția `mysql_num_fields()` pentru a determina numărul coloanelor dintr-un set de rezultate care conțin rândurile selectate folosind specificatorul SQL pentru câmpuri \*:

```
$interogare = "SELECT * FROM angajat";
$resultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
$numar_campuri = mysql_num_fields($resultat);
```

## Obținerea numelui unei coloane din setul de rezultate

Funcția `mysql_field_name()` returnează numele coloanei din setul de rezultate având valoarea indexului dată ca argument al funcției. Indexul asociat cu prima coloană este 0, indexul asociat celei de-a doua coloane este 1 etc.

De exemplu, programul următor folosește funcția `mysql_field_name()` pentru a determina numele primei coloane din setul de rezultate:

```
$interogare = "SELECT * FROM angajat";
$resultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
$nume = mysql_field_name($resultat, 0);
```

## Obținerea lungimii unei coloane dintr-un set de rezultate

Funcția `mysql_field_len()` returnează lungimea maximă a coloanei dintr-un set de rezultate, având valoarea indexului dată ca argument al funcției. Indexul asociat primei coloane este 0, indexul asociat celei de-a doua coloane este 1 etc.

De exemplu, programul următor folosește funcția `mysql_field_len()` pentru a determina lungimea maximă a primei coloane din setul de rezultate:

```
$interogare = "SELECT * FROM angajat";
$resultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
$lungime = mysql_field_len($resultat, 0);
```

## Obținerea indicatorilor MySQL asociați unei coloane a setului de rezultate

Funcția `mysql_field_flags()` returnează indicatorii SQL asociați coloanei din setul de rezultate al cărei index este dat ca argument al funcției. Indexul asociat primei coloane este 0, indexul asociat celei de-a doua coloane este 1 etc. Funcția `mysql_field_flags()` raportează următorii indicatori:

- `AUTO_INCREMENT`
- `BINARY`
- `BLOB`
- `ENUM`
- `MULTIPLE_KEY`
- `NOT_NULL`
- `PRIMARY_KEY`
- `TIMESTAMP`
- `UNIQUE_KEY`
- `UNSIGNED`
- `ZEROFILL`

Dacă la o coloană sunt asociați mai mulți indicatori, fiecare indicator este separat de vecinii săi prin intermediul unui singur spațiu.

De exemplu, programul următor folosește funcția `mysql_field_flags()` pentru a determina indicatorii asociați primei coloane din setul de rezultate:

```
$interogare = "SELECT * FROM angajat";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": " . mysql_error()."
");
}
$indicatori = mysql_field_flags($rezultat, 0);
```

## Obținerea tipului MySQL al unei coloane din setul de rezultate

Funcția `mysql_field_type()` returnează tipul MySQL al unei coloane din setul de rezultate, coloană al cărei index este dat ca argument al funcției. Indexul asociat primei coloane este 0, indexul asociat celei de-a doua coloane este 1 etc. Tabelul 13-1 descrie principalele tipuri MySQL returnate de această funcție.

De exemplu, programul următor folosește funcția `mysql_field_type()` pentru a determina tipul primei coloane din setul de rezultate:

```
$interogare = "SELECT * FROM angajat";
$rezultat = mysql_query($interogare);
if (mysql_errno())
```

```
{
 die("
" . mysql_errno().": " . mysql_error()."
");
}
$tip = mysql_field_type($rezultat, 0);
```

## Determinarea tabelului MySQL asociat unei coloane din setul de rezultate

Funcția `mysql_field_table()` returnează tabelul MySQL, dacă există, asociat coloanei din setul de rezultate al cărei index este dat de argumentul funcției. Indexul asociat primei coloane este 0, indexul asociat celei de-a doua coloane este 1 etc. În cazul în care coloana conține o valoare calculată sau dacă respectiva coloană nu este asociată în alt mod cu un tabel MySQL, funcția returnează un șir vid.

De exemplu, programul următor folosește funcția `mysql_field_table()` pentru a determina tabelul asociat primei coloane din setul de rezultate:

```
$interogare = "SELECT * FROM angajat";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": " . mysql_error()."
");
}
$tabel = mysql_field_table($rezultat, 0);
```

## Obținerea structurii complete a setului de rezultate

Dacă sunteți interesat în obținerea mai multor caracteristici ale setului de rezultate, funcția `mysql_fetch_field()` poate fi utilă. Această funcție returnează un obiect ale cărui proprietăți conțin o varietate de informații cu privire la coloana unui set de rezultate. Proprietățile sunt următoarele:

- `blob` are valoarea 1 în cazul în care coloana este de tip BLOB
- `max_length` – lungimea maximă a coloanei;
- `multiple_key` are valoarea 1 în cazul în care coloana este o cheie non-unică
- `name` – numele coloanei
- `not_null` are valoarea 1 în cazul în care coloana nu poate conține valoarea NULL
- `numeric` are valoarea 1 în cazul în care coloana este numerică
- `primary_key` are valoarea 1 în cazul în care coloana este o cheie primară
- `table` – numele tabelului MySQL căruia îi aparține coloana
- `type` – tipul MySQL al coloanei
- `unique_key` are valoarea 1 în cazul în care coloana este o cheie unică
- `unsigned` are valoarea 1 în cazul în care coloana este de tip UNSIGNED
- `zerofill` are valoarea 1 în cazul în care coloana este completată cu zerouri

Ca și funcția conexă descrisă anterior în acest modul, funcția `mysql_fetch_field()` preia două argumente: valoarea returnată de funcția `mysql_query()` și indexul



coloanei din setul de rezultate care va fi descrisă. Ca de obicei, indexul asociat primei coloane este 0, indexul asociat celei de-a doua coloane este 1 etc.

Iată un exemplu care prezintă modul de obținere a structurii complete a setului de rezultate, inclusiv o descriere a fiecărei coloane din setul de rezultate:

```
$interogare = "SELECT * FROM angajat";
$rezultat = mysql_query($interogare);
if (mysql_errno())
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
$numar_campuri = mysql_num_fields($rezultat);
for ($i = 0; $i < $numar_campuri; $i++)
{
 echo "
Coloana $i: ";
 $info = mysql_fetch_field($rezultat);
 if ($info)
 {
 echo "<PRE>";
 blob: $info->blob
 max_length: $info->max_length
 multiple_key: $info->multiple_key
 name: $info->name
 not_null: $info->not_null
 numeric: $info->numeric
 primary_key: $info->primary_key
 table: $info->table
 type: $info->type
 unique_key: $info->unique_key
 unsigned: $info->unsigned
 zerofill: $info->zerofill
 </PRE>";
 }
 else
 {
 echo "Necunoscut";
 }
}
```

## Accesul non-secvențial la coloanele unui set de rezultate

Funcțiile `mysql_fetch_row()` și `mysql_fetch_array()` returnează, în general, rândurile dintr-un set de rezultate în mod secvențial. Totuși, funcția `mysql_data_seek()` permite obținerea accesului la rândurile unui set de rezultate într-o manieră non-secvențială. Funcția are forma:

```
mysql_data_seek(rezultat, numar_rand)
```

unde *rezultat* este valoarea returnată de funcția `mysql_query()`, iar *numar\_rand* este indexul rândului la care doriți să obțineți accesul. Primul rând al setului de rezultate este numerotat cu 1, al doilea cu 2 etc. Funcția returnează `true` dacă execuția reușește, respectiv `false` în caz contrar. O invocare ulterioară a funcției

`mysql_fetch_row()` sau a funcției `mysql_fetch_array()` va returna rândul din poziția specificată a setului de rezultate.

De exemplu, următorul program obține accesul la al doilea rând al setului de rezultate returnat de o interogare anterioară:

```
$ok = mysql_data_seek($rezultat, 1);
if (!$ok)
{
 die("
" . mysql_errno().": ".mysql_error()."
");
}
$rand = mysql_fetch_array($rezultat, MYSQL_ASSOC);
```



### Test „la minut”

- Precizați funcția MySQL din biblioteca PHP care returnează numele unei coloane specificate dintr-un set de rezultate.
- Precizați funcția MySQL din biblioteca PHP care returnează tipul MySQL al unei coloane specificate dintr-un set de rezultate.
- Precizați funcția MySQL din biblioteca PHP care returnează numărul coloanelor dintr-un set de rezultate.
- Precizați funcția MySQL din biblioteca PHP care permite accesul non-secvențial la un set de rezultate.

## Explorarea SGBD

Biblioteca MySQL aferentă limbajului PHP include trei funcții care vă permit să determinați structura unei baze de date, în speță:

- Să determinați bazele de date găzduite de serverul MySQL
- Să determinați tabelele incluse într-o bază de date specificată
- Să determinați coloanele incluse într-un tabel specificat dintr-o bază de date, precum și caracteristicile acestor coloane



### Sfatul specialistului

**Întrebare:** Dacă am scris interogarea care a returnat setul de rezultate, n-ar trebui să cunosc deja structura acestuia? De ce trebuie să invoc o funcție pentru a determina, de exemplu, numele unei coloane din setul de rezultate?

Răspunsuri la test:

- `mysql_field_name();`
- `mysql_field_type();`
- `mysql_num_fields();`
- `mysql_data_seek();`



**Răspuns:** Există numeroase circumstanțe în care funcțiile descrise în această secțiune se pot dovedi utile. O asemenea situație apare când formați o interogare folosind forma `SELECT *`. Ordinea în care este aranjat conținutul unui set de rezultate nu este definită și poate varia în funcție de versiunea MySQL pe care o utilizați. De asemenea, conținutul setului de rezultate se va modifica la fiecare schimbare în structura tabelului supus interogării. Un alt exemplu de situație în care funcțiile respective sunt utile este o aplicație care permite utilizatorilor să emită interogări sau să furnizeze date folosite la formarea interogărilor. Într-un asemenea caz, structura setului de rezultate nu este cunoscută în momentul scrierii programului PHP.

## Determinarea bazelor de date găzduite de un server

Pentru a determina bazele de date găzduite de un server MySQL de care programul dumneavoastră este legat prin intermediul unei conexiuni active, invocați funcția:

```
mysql_list_dbs()
```

Funcția returnează un set de rezultate special, alcătuit din numele bazelor de date găzduite. Puteți determina numărul rândurilor din setul de rezultate invocând funcția `mysql_num_rows()`, așa cum procedați în cazul unui set de rezultate normal. Cu toate acestea, trebuie să preluați rândurile folosind funcția `mysql_tablename()`, care preia ca argumente valoarea returnată de funcția `mysql_list_dbs()` și numărul rândurilor care vor fi preluate. Rândurile sunt numerotate începând de la 0.

Iată un exemplu care prezintă modul de afișare a numelor bazelor de date găzduite:

```
$db = mysql_connect("localhost");
$dblist = mysql_list_dbs();
$n = mysql_num_rows($dblist);
for ($i = 0; $i < $n; $i++)
{
 echo "
" . mysql_tablename($dblist, $i);
}
```

Din motive de claritate, în exemplu nu se verifică apariția unor eventuale erori MySQL; înainte de a folosi acest program, trebuie să adăugați instrucțiuni adecvate de verificare a apariției erorilor.

## Notă

PHP2 conținea o funcție, numită `mysql_dbname()`, folosită pentru regăsirea numelor bazelor de date din structura de date returnată de `mysql_list_dbs()`. Totuși, în PHP4, pentru această operație trebuie folosită funcția `mysql_tablename()`.

## Determinarea tabelelor incluse într-o bază de date

Pentru a obține o listă a tabelelor incluse într-o bază de date specificată, invocați funcția `mysql_list_tables()`, transferându-i ca argument numele bazei de date. Programul dumneavoastră trebuie să dispună de o conexiune activă cu serverul MySQL; în caz contrar, funcția eșuează. Funcția `mysql_list_tables()` returnează un set de rezultate special, similar celui returnat de `mysql_list_dbs()`. Pentru a obține acces la lista cu tabele, parcurgeți prin iterație setul de rezultate, invocând în mod repetat funcția `mysql_tablename()`.

Iată un exemplu care prezintă modul de obținere și afișare a listei tabelelor asociate bazei de date `testdb`:

```
$db = mysql_connect("localhost", "php", "salut");
$tabele = mysql_list_tables("testdb");
$n = mysql_num_rows($tabele);
for ($i = 0; $i < $n; $i++)
{
 echo "
" . mysql_tablename($tabele, $i);
}
```

Din motive de claritate, în exemplu nu se verifică apariția unor eventuale erori MySQL; totuși, înainte de a folosi acest program, trebuie să adăugați instrucțiuni adecvate de verificare a apariției erorilor.

## Sugestie

În cazul în care invocarea funcției `mysql_num_rows()` eșuează cu mesajul „Warning: Supplied argument is not a valid MySQL result resource”, probabil că identificatorul de utilizator sau parola specificate la invocarea funcției `mysql_connect()` nu au permisiunea de a obține acces la baza de date ale cărei tabele încercați să le afișați.

## Determinarea coloanelor incluse într-un tabel

Pentru a obține o listă a coloanelor incluse într-un tabel, invocați funcția `mysql_list_fields()`, transferându-i ca argument numele bazei de date și numele tabelului. Programul dumneavoastră trebuie să dispună de o conexiune activă cu serverul MySQL; în caz contrar, funcția eșuează. Funcția `mysql_list_fields()` returnează un set de rezultate similar celor returnate de funcțiile `mysql_list_dbs()` și `mysql_list_tables()`. Pentru a obține acces la lista coloanelor și la caracteristicile acestora, invocați funcția `mysql_fetch_fields()`. Alternativ, dacă doriți să obțineți acces la o singură caracteristică a coloanelor, puteți invoca una din funcțiile `mysql_field_flags()`, `mysql_field_len()`, `mysql_field_name()` sau `mysql_field_type()`.

\* În traducere Avertisment: Argumentul furnizat nu constituie o resursă de tip rezultat MySQL corectă. – N.T.

Iată un exemplu care vă prezintă modul de obținere a listei coloanelor și a caracteristicilor coloanelor din tabelul angajat al bazei de date testdb:

```
$db = mysql_connect("localhost", "php", "salut");
$campuri = mysql_list_fields("testdb", "angajat");
$numar_campuri = mysql_num_fields($campuri);
for ($i = 0; $i < $numar_campuri; $i++)
{
 echo "
Coloana $i: ";
 $info = mysql_fetch_field($campuri);
 if ($info)
 {
 echo "<PRE>
 blob: $info->blob
 max_length: $info->max_length
 multiple_key: $info->multiple_key
 name: $info->name
 not_null: $info->not_null
 numeric: $info->numeric
 primary_key: $info->primary_key
 table: $info->table
 type: $info->type
 unique_key: $info->unique_key
 unsigned: $info->unsigned
 zerofill: $info->zerofill
 </PRE>";
 }
 else
 {
 echo "Necunoscut";
 }
}
```

Din motive de claritate, în exemplu nu se verifică apariția unor eventuale erori MySQL. Din nou, înainte de a folosi acest program, trebuie să adăugați instrucțiuni adecvate de verificare a apariției erorilor.

### Sugestie

În cazul în care invocarea funcției `mysql_list_fields()` eșuează cu mesajul de eroare „1044: Access denied”, probabil că identificatorul de utilizator sau parola specificate la invocarea funcției `mysql_connect()` nu au permisiunea de a obține acces la tabelul din baza de date ale cărei coloane încercați să le afișați.



### Sfatul specialistului

**Întrebare:** Pot rula programul `mysql` în mod interactiv, pentru a vizualiza structura unei baze de date MySQL. Care ar fi, atunci, motivul pentru care aș folosi funcțiile descrise în această secțiune?

**Răspuns:** Funcțiile descrise în această secțiune vă permit să determinați structura unei baze de date la rulare. Le puteți folosi, de exemplu, pentru a crea

\* În traducere: *Acces interzis.* – N.T.

utilitare care permit unui utilizator să modifice structura unei baze de date în mod interactiv. De asemenea, le puteți folosi pentru a crea formulare ce permit utilizatorilor care nu cunosc SQL să formeze interogări SQL care raportează datele incluse într-o bază de date MySQL.



### Test „la minut”

- Precizați numele funcției MySQL din biblioteca PHP care afișează numele bazelor de date găzduite de un server.
- Precizați numele funcției MySQL din biblioteca PHP care afișează numele tabelelor dintr-o bază de date MySQL.
- Precizați numele funcției MySQL din biblioteca PHP care menționează coloanele dintr-un tabel al unei baze de date MySQL.

## Ghilimele și ghilimele magice

Să presupunem că încercați să executați o interogare MySQL care este asemănătoare cu următoarea:

```
SELECT * FROM tabel WHERE text="Ce este asta?" intreba ea';
```

Dincolo de alte aspecte, veți întâmpina unele dificultăți, deoarece SQL nu permite înglobarea unor ghilimele simple în interiorul valorii unui șir. Probleme similare pot apărea când un utilizator neatenț sau cu intenții rele tastează un text ca acesta într-o casetă cu text a unui formular HTML:

```
<INPUT TYPE="TEXT" NAME="parola">
```

Dacă emiteți o instrucțiune de reflectare a conținutului casetei text fără să vă gândiți, veți descoperi că pagina HTML rezultantă conține o casetă cu text nedorită.

PHP include funcții și facilități pentru rezolvarea acestor situații. Această secțiune prezintă unele tehnici pentru lucrul cu date utilizate la:

- Interogări SQL
- Pagini HTML
- Adrese URL

## Ghilimele magice

Fișierul de inițializare PHP, în speță `php.ini`, conține opțiuni de configurare care controlează modul în care PHP controlează datele provenite de la o sursă externă, cum sunt un formular HTML, un fișier text sau o bază de date. Aceste opțiuni sunt

Răspunsuri la test:

- `mysql_list_dbs()`
- `mysql_list_tables()`
- `mysql_list_fields()`

proiectate pentru a vă ajuta să vă adaptați la modalitățile deseori contradictorii în care browserele și bazele de date manipulează caracterele speciale. În general, administratorul de sistem configurează fișierul `php.ini` atunci când este instalat PHP; în general, utilizatorii obișnuiți nu trebuie să aibă posibilitatea de a aduce modificări în fișier.

Opțiunea `magic_quotes_gpc` specifică modul în care PHP manipulează operațiile HTTP GET și POST, precum și operațiile cu variabilele cookie. Dacă opțiunea este activată, PHP ignoră în mod automat ghilimelele simple, ghilimelele duble, caracterele backslash și caracterele nule (caracterele a căror valoare ASCII este 0) care apar într-o variabilă HTTP, prefixând fiecare apariție a acestor caractere cu un caracter backslash. În mod caracteristic, această opțiune este activată într-o instalare prestabilită.

Opțiunea `magic_quotes_runtime` specifică modul în care PHP manipulează datele de origine externă. Dacă această opțiune este activată, PHP ignoră automat ghilimelele simple și duble care apar în datele externe, inclusiv datele din formularele HTML, din fișiere și baze de date. În mod caracteristic, această opțiune este dezactivată într-o instalare prestabilită.

Dacă este activată, opțiunea `magic_quotes_sybase` modifică efectul opțiunilor `magic_quotes_gpc` și `magic_quotes_runtime`, astfel încât un caracter de tip ghilimele simple să fie prefixat cu un alt caracter de tip ghilimele simple, nu cu un backslash. Această opțiune respectă convențiile neobișnuite de manipulare a șirurilor, folosite de sistemul Sybase de gestiune a bazelor de date. În mod caracteristic, această opțiune este dezactivată într-o instalare prestabilită.

Funcțiile `get_magic_quotes_gpc()` și `get_magic_quotes_runtime()` returnează fiecare valoarea opțiunii PHP corespunzătoare. Mai mult, puteți folosi funcția `set_magic_quotes_runtime()` pentru a specifica valoarea opțiunii `magic_quotes_runtime` pentru restul duratei scriptului PHP curent. De exemplu, pentru a dezactiva opțiunea `magic_quotes_runtime`, invocați funcția

```
set_magic_quotes_runtime(0);
```

Alternativ, pentru a activa opțiunea `magic_quotes_runtime`, invocați funcția

```
set_magic_quotes_runtime(1);
```

PHP nu furnizează nici o funcție corespunzătoare care să anuleze opțiunea `magic_quotes_runtime`, deoarece opțiunea respectivă este utilizată la transferul variabilelor HTTP către un script. După ce scriptul și-a început execuția, valoarea opțiunii respective nu mai este luată din nou în considerare.

## Conversia caracterelor speciale și anularea acestora

Când PHP convertește\* un șir, folosește pentru aceasta funcția `addslashes()`. Dacă PHP nu este configurat astfel încât să convertească un text în mod automat, atunci dumneavoastră trebuie să executați manual această modificare. Pur și simplu invocați funcția `addslashes()`, transferându-i ca argument șirul text. De exemplu, pentru a converti conținutul șirului `$text` și a plasa rezultatul în variabila `$modificat`, invocați funcția după cum urmează:

```
$modificat = addslashes($text);
```

Funcția `stripslashes()` execută operația complementară de anulare a conversiei, adică elimină caracterele backslash inserate pentru conversia caracterelor de tip ghilimele simple, ghilimele duble, backslash și caractere nule. Pentru a anula conversia șirului `$modificat` și a plasa rezultatul în variabila `$nemodificat`, invocați funcția după cum urmează:

```
$nemodificat = stripslashes($modificat);
```

### Sugestie

PHP mai include o funcție utilă pentru conversia caracterelor speciale. Funcția `quotemeta()` va insera un backslash înainte de fiecare apariție a următoarelor caractere: `. \ + * ? [ ^ ] ( $ )`.

## Conversia textelor HTML

Când lucrați cu texte HTML, conversia executată de funcția `addslashes()` nu este suficientă, deoarece HTML este sensibil la alte caractere speciale decât cele convertite de funcția `addslashes()`. Funcția `htmlspecialchars()` convertește caracterele la care HTML este sensibil. Această funcție este utilă pentru a exista garanția că un anumit șir nu conține marcaje HTML, ceea ce poate fi important pentru a corecta modul de operare a unei aplicații, cum ar fi o tabelă de mesaje Web. Funcția execută următoarele conversii:

Caracter	Rezultat convertit
& (ampersand)	&amp;
" (ghilimele duble)	&quot; (numai când este specificată opțiunea ENT_COMPAT sau ENT_QUOTES)
' (ghilimele simple)	&#39; (numai când este specificată opțiunea ENT_QUOTES)
< (mai mic decât)	&lt;
> (mai mare decât)	&gt;

\* Această conversie nu trebuie confundată cu o alta, și anume conversia de tip. Conversia menționată în acest capitol constă, de fapt, din modificarea semnificației anumitor caractere, pentru a preveni o interpretare eronată a acestora de către programele specializate. Din acest motiv, toate conversiile menționate în acest capitol vor face referire la modificarea semnificației unor caractere, nu la conversiile de tip. – N.T.

Funcția are următoarea formă:

```
htmlspecialchars(text [,stil_ghilimele])
```

unde *text* specifică șirul care conține textul ce urmează a fi convertit, iar argumentul opțional *stil\_ghilimele* specifică modul de conversie a ghilimelelor simple, respectiv duble. Valorile permise ale opțiunii *stil\_ghilimele* sunt

- ENT\_COMPAT precizează că trebuie convertite numai ghilimelele duble
- ENT\_QUOTES precizează că trebuie convertite ambele tipuri de ghilimele
- ENT\_QUOTES precizează că nu trebuie convertit nici un tip de ghilimele

De exemplu, pentru a converți toate caracterele HTML speciale, inclusiv ambele tipuri de ghilimele, din șirul *\$htmltext*, invocați funcția după cum urmează:

```
$convertit = htmlspecialchars($htmltext);
```

### Sugestie

O funcție PHP conexasă, și anume *htmlspecialchars()*, convertește toate caracterele cu echivalente în entități HTML. În prezent, nu este acceptat decât setul de caractere ISO-8859-1.

### Sugestie

PHP include o altă funcție utilă pentru conversia textelor HTML. Funcția *n2br()* inserează caracterele *<br>* anterior fiecărui caracter de tip linie nouă din argumentul său. Începând de la PHP 4.0.5, funcția inserează caracterele *<br/>*, în concordanță cu XHTML 1.0.

## Conversia adreselor URL

Când PHP codifică date ca parte a unei adrese URL, așa cum procedează la efectuarea unei operații HTTP GET, folosește funcția *urlencode()*. Această funcție înlocuiește fiecare caracter non-alfanumeric (cu excepția spațiilor) cu un simbol al procentului (%), urmat de două cifre hexazecimale care conțin valoarea ASCII a caracterului. Spațiile sunt codificate sub formă de simboluri ale adunării (+).

O funcție conexasă, în speță *rawurlencode()*, execută o conversie similară; cu toate acestea, funcția respectivă înlocuiește spațiile cu *%20*, conform RFC 1738, standardul Internet pentru codificarea adreselor URL (vezi [www.rfc.net](http://www.rfc.net)).

Funcțiile *urldecode()* și *rawurldecode()* execută operații complementare.

Aceste funcții sunt utile la crearea adreselor URL care includ perechi nume-valoare înglobate. De exemplu, să presupunem că scriptul dumneavoastră PHP trebuie să creeze o pagină HTML care conține o hiperlegătură spre adresa URL a unui motor de căutare. Termenul de căutare trebuie transmis în formă codificată URL, pentru ca serverul Web și browserul să nu se „împiedice” de caracterele speciale. Pentru a codifica datele, folosiți un program PHP ca acesta:

```
echo "A HREF='www.motordecutare.com?cuvantcheie=' .
urlencode ($text). ">";
```



## Sfatul specialistului

**Întrebare:** Sunt administratorul unui server PHP. Care sunt opțiunile pentru ghilimele magice pe care ar trebui să le activez?

**Răspuns:** Opțiunile adecvate constituie obiectul unor oarecare controverse. Interesele programatorilor PHP începători, care poate că nu înțeleg în totalitate necesitatea încadrării datelor între ghilimele, sunt cel mai bine reprezentate prin activarea opțiunilor *magic\_quotes\_gpc* și *magic\_quotes\_runtime*. În acest caz, opțiunea *magic\_quotes\_sybase* trebuie activată numai dacă principalul sistem de gestiune a bazelor de date folosit cu PHP este Sybase.

Pe de altă parte, mulți programatori PHP pricepuți sunt de părere că ghilimele magice sunt o mare pacoste. Dacă serverul dumneavoastră PHP este folosit mai ales de către programatori PHP experimentați, trebuie să aveți în considerare dezactivarea opțiunilor legate de ghilimelele magice.



### Test „la minut”

- Care este funcția PHP folosită de către ghilimelele magice din PHP pentru încadrarea între ghilimele a datelor provenite din surse externe?
- Care este funcția PHP ce poate fi utilizată pentru a elimina caracterele backslash adăugate prin caracteristica ghilimelelor magice?
- Care este opțiunea de configurare PHP ce influențează datele transmise prin intermediul operațiilor HTTP GET și POST?

## Alte facilități PHP de gestiune a datelor

Biblioteca PHP furnizează acces la numeroase alte instrumente de gestiune a datelor. Secțiunea de față trece în revistă aceste instrumente, le descrie caracteristicile și situațiile când își pot dovedi utilitatea. Această secțiune nu este menită a prezenta informațiile necesare pentru utilizarea instrumentelor descrise. În schimb, furnizează referințe la cărți sau situri Web care vă pot ajuta să învățați mai multe despre fiecare instrument. Instrumentele prezentate nu sunt disponibile într-o instalare PHP prestabilită; în general, administratorul dumneavoastră PHP trebuie să le instaleze și să le configureze separat din PHP.

Răspunsuri la test:

- *addslashes()*
- *stripslashes()*
- *magic\_quotes\_gpc*

## Postgresql

Ca și MySQL, Postgresql (pronunțat POST-gres-q-l), este un sistem de gestiune a bazelor de date provenit dintr-o sursă deschisă. Din perspectivă istorică, MySQL a fost sistemul de baze de date preferat de amatorii de performanțe brute și simplitate în utilizare, în timp ce Postgresql a fost opțiunea favorită a celor care preferă caracteristicile mai complexe și scalabilitatea. Totuși, versiunile recente ale acestor sisteme au complicat problema, deoarece MySQL a acumulat unele caracteristici mai complexe, iar performanțele sistemului Postgresql s-au îmbunătățit.

Cu toate acestea, Postgresql continuă să accepte numeroase caracteristici care nu sunt încă oferite de versiunile MySQL standard, între care amintim:

- **Chei externe.** Acestea permit respingerea automată a modificărilor din baza de date care nu respectă structura bazei de date.
- **Subselecții.** Acestea permit formarea unor interogări complexe, care reduc la minimum dimensiunea seturilor de rezultate trimise prin rețea.
- **Tranzacții.** Acestea permit evitarea aplicării incomplete asupra unei baze de date a unui set de modificări corelate, precum și alterarea datelor care derivă din procesul respectiv.
- **Declanșatoare.** Permit specificarea de acțiuni executate de serverul de baze de date ori de câte ori se produc anumite evenimente.
- **Vederi.** Permit furnizarea, în condiții convenabile, către anumiți utilizatori, de acces la sub-seturi ale bazei de date.

Mai mult, Postgresql pare să se adapteze mai bine decât MySQL la aplicații care implică mai mulți utilizatori concurenți ai bazei de date. Dacă sunteți interesat de Postgresql, consultați situl Web aferent acestuia, la adresa [www.postgresql.org](http://www.postgresql.org).

### Sugestie

Utilizatorii de Linux Red Hat 7.1 pot instala suportul PHP pentru Postgresql prin instalarea pachetului RPM `php-pgsql`.

## ODBC

ODBC (Open Database Connectivity) este un standard creat de Microsoft, acum adoptat pe scară largă atât în mediile Microsoft, cât și în cele bazate pe UNIX/Linux. ODBC furnizează o facilitare minimală pentru accesul la bazele de date. În esență, fiecare sistem comercial de gestiune a bazelor de date și majoritatea sistemelor non-comerciale de gestiune a bazelor de date furnizează programe driver ODBC care permit accesul la aceste sisteme prin intermediul ODBC, nu numai prin intermediul unor programe driver native. ODBC este deosebit de important pentru cei care doresc să obțină accesul la o bază de date găzduită de Microsoft dintr-un sistem UNIX/Linux.

Sistemele de operare create de Microsoft conțin suport încorporat pentru ODBC. Astfel, dacă rulați PHP sub un sistem de operare Microsoft, veți descoperi că suportul ODBC este disponibil instantaneu. Totuși, sub UNIX/Linux trebuie să instalați o punte ODBC-ODBC sau o altă facilitare care acceptă ODBC. Soluțiile comerciale le includ pe cele distribuite de Openlink ([www.openlinksw.com](http://www.openlinksw.com)) și Easysoft ([www.easysoft.com](http://www.easysoft.com)). O alternativă provenită dintr-o sursă deschisă, în speță ODBCsocketServer, este disponibilă la adresa [sourceforge.net](http://sourceforge.net).

## LDAP

Protocolul Lightweight Directory Access Protocol (LDAP) devine tot mai popular. Ca o bază de date obișnuită, o bază de date LDAP stochează date. O bază de date LDAP este însă foarte bine adaptată pentru stocarea cataloagelor, adică a unor liste cu persoane și unități de organizare, precum și a caracteristicilor acestora. Bazele de date LDAP sunt proiectate pentru a stoca date relativ simple, rareori modificate. Astfel, acestea sunt ideale pentru stocarea informațiilor despre utilizatorii sistemului. În particular, LDAP acceptă metode de comunicare sigure, care permit bazelor de date LDAP să stocheze nume de utilizator și parole. LDAP furnizează mecanisme pentru replicarea bazelor de date LDAP în vederea protecției integrității datelor și asigură echilibrarea încărcării, pentru a putea tolera sarcini de prelucrare de mari dimensiuni.

Pentru mai multe informații despre LDAP, consultați volumul *Implementing Directory Services* (Implementarea serviciilor de catalog), de Archie Reed (Osborne/McGraw-Hill, 2000).

### Sugestie

Dacă folosiți Red Hat Linux 7.1, puteți instala suportul PHP pentru LDAP prin instalarea pachetului RPM `php-pgsql`. Apoi, modificați fișierul `/etc/php.ini` astfel încât să conțină linia (ne-comentată) `extension=ldap.so`. Apoi, reporniți serviciul HTTP prin emiterea comenzii `service httpd restart`.

## XML

Extensible Markup Language (XML) este un limbaj pentru descrierea datelor. Mulți se așteaptă ca, într-un târziu, XML să înlocuiască HTML ca limbaj dominant pentru schimbul informațiilor prin Internet. În timp ce HTML descrie aspectul datelor, XML este capabil de a descrie structura datelor (practic, XML este o modalitate de a trimite o mică bază de date prin Internet).

PHP acceptă biblioteca `xpath`, bazată pe activitatea lui James Clark. Biblioteca vă permite să construiți un analizor pentru documentele XML. Un analizor înțelege sintaxa unui document XML și poate identifica acele componente structurale care



alcătuiesc documentul. Asociind o funcție cu fiecare tip de componentă, puteți configura analizorul astfel încât să prelucreze sau să convertească un document XML.

## ? Sfatul specialistului

**Întrebare:** Care sunt celelalte instrumente și caracteristici acceptate de PHP?

**Răspuns:** Bibliotecile PHP sunt extrem de cuprinzătoare. Alte instrumente de gestiune a datelor acceptate de PHP includ următoarele:

- dBase
- DBM
- dbx
- DOM XML
- FrontBase
- filePro
- Hyperwave Information Server
- Informix
- Interbase
- Ingres II
- Microsoft SQL Server
- mSQL
- Oracle și Oracle 8
- Extensia CORBA Satellite
- SESAM/SQL Server
- Sybase
- WDDX
- YAZ (protocolul Z39.50)

Pentru mai multe informații despre aceste caracteristici ale limbajului PHP, consultați referințele prezentate în manualul PHP pe suport electronic, la adresa <http://www.php.net>.

Pentru mai multe informații despre XML, consultați pagina Web a organizației World Wide Web Consortium (<http://www.w3.org/XML/>). De asemenea, consultați volumul *XML: The Complete Reference* (XML – o referință completă), de Heather Williamson (Osborne/McGraw-Hill, 2001).

### Sugestie

Dacă folosiți Red Hat Linux 7.1, pachetul PHP standard include suport pentru XML.



### Test „la minut”

- Care este sistemul de gestiune a bazelor de date provenit din sursă deschisă (altul decât MySQL) frecvent folosit cu PHP?
- Care este serviciul de cataloage frecvent folosit cu PHP?
- Care este caracteristica PHP ce furnizează accesul la sistemele de baze de date Microsoft?
- Care este limbajul ce va înlocui, în cele din urmă, HTML ca limbaj comun al Internetului?



### Proiect 14-1: Un program de navigare prin agenda cu adrese

Acest proiect abordează din nou programul de navigare prin agenda cu adrese prezentat în cadrul proiectului 11-3. În acel proiect, programul de navigare era implementat prin utilizarea fișierelor. Proiectul de față va implementa un program de navigare similar, care folosește o bază de date MySQL pentru a stoca nume și adrese de e-mail.

#### Scopurile proiectului

- Prezentarea unui program complet care obține accesul la o bază de date MySQL
- Prezentarea procedeelelor de programare a bazelor de date care acceptă navigarea

#### Pas cu pas

1. Creați următorul script de shell, denumindu-l `p14-1.sh`. Modificați în mod corespunzător identificatorul de utilizator (`php`) și parola (`salut`). Apoi, încărcați scriptul în serverul dumneavoastră PHP.

```
mysql -p <<EOF
#DROP DATABASE maildb;
CREATE DATABASE maildb;
USE maildb;
GRANT ALL ON maildb.* TO php@localhost IDENTIFIED BY 'salut';
CREATE TABLE agenda_adrese
(
 id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
 nume VARCHAR(50),
 email VARCHAR(50)
)
INSERT INTO agenda_adrese
(nume, email)
VALUES
('Al Nall', 'al@browncow.com');
```

#### Răspunsuri la test:

- Postgresql
- LDAP
- ODBC
- XML



```
('Bob Tale', 'bob@stories.com'),
('Chuck Stake', 'chuck@beef.com'),
('Ed Nogg', 'ed@beverage.com'),
('Xi Lentz', 'xi@quiet.com'),
('Yo Hoho', 'yo@pirates.com'),
('Zak Cloth', 'zak@ashes.com');
EOF
```

2. Studiați scriptul, pentru a înțelege modul de funcționare a acestuia. Rețineți că scriptul creează o bază de date numită maildb, precum și un tabel agenda\_adrese în interiorul acesteia. De asemenea, scriptul populează tabelul prin inserția mai multor rânduri. Linia care începe cu un simbol diez (#) este un comentariu. Dacă eliminați simbolul diez și executați scriptul, acesta va șterge și apoi va re-crea baza de date. Trebuie să ștergeți simbolul diez înainte de a rula scriptul a doua oară; în caz contrar, scriptul va eșua în încercarea de a crea baza de date, care există deja.
3. Deschideți sesiunea de lucru cu gazda serverului PHP, treceți la catalogul care conține scriptul încărcat și executați scriptul, prin emiterea următoarei comenzi:

```
sh p14-1.sh
```

4. Creați următorul script PHP, denumindu-l p14-1.php. Ca mai înainte, modificați în mod corespunzător identificatorul de utilizator și parola. Încărcați scriptul în serverul dumneavoastră PHP, plasându-l într-un catalog adecvat pentru a fi accesibil din Web:

```
<HTML>
<HEAD>
<TITLE>Program de navigare în agenda cu adrese</TITLE>
</HEAD>
<BODY>
<H2>Program de navigare în agenda cu adrese</H2>
<FORM METHOD="POST" ACTION="p14-1.php">
<?php
funcțion check_mysql()
{
 if (mysql_errno() > 0)
 {
 die("
Eroare MySQL " . mysql_errno() . ": " . mysql_error());
 }
}

$db = mysql_connect("localhost", "php", "salut");
if (!$db)
{
 die("Nu s-a putut deschide conexiunea cu serverul MySQL.");
}

mysql_select_db("maildb");
check_mysql();

if (!isset($id))
{
 $id = 0;
}

if (isset($stanga))
```

```
{
 $interogare = "SELECT id, nume, email FROM agenda_adrese "
 . "WHERE id < $id ORDER BY id DESC";
 $rezultat = mysql_query($interogare);
 check_mysql();
 $rand = mysql_fetch_row($rezultat);
 check_mysql();
 if ($rand[0] > 0)
 {
 $id = $rand[0];
 $nume = $rand[1];
 $email = $rand[2];
 }
 elseif (isset($dreapta))
 {
 $interogare = "SELECT id, nume, email FROM agenda_adrese "
 . "WHERE id > $id ORDER BY id ASC";
 $rezultat = mysql_query($interogare);
 check_mysql();
 $rand = mysql_fetch_row($rezultat);
 check_mysql();
 if ($rand[0] > 0)
 {
 $id = $rand[0];
 $nume = $rand[1];
 $email = $rand[2];
 }
 }
 elseif (isset($cauta))
 {
 $id = 0;
 $interogare = "SELECT id, nume, email FROM agenda_adrese "
 . "WHERE nume LIKE '%" . $nume . "%' AND id > $id";
 $rezultat = mysql_query($interogare);
 check_mysql();
 $rand = mysql_fetch_row($rezultat);
 check_mysql();
 if ($rand[0] > 0)
 {
 $id = $rand[0];
 $nume = $rand[1];
 $email = $rand[2];
 }
 }
 elseif (isset($adauga))
 {
 $interogare = "INSERT INTO agenda_adrese "
 . "(nume, email) VALUES ('" . $nume . "', '" . $email . "')";
 $rezultat = mysql_query($interogare);
 check_mysql();
 $id = mysql_insert_id();
 $mesaj = "A fost adaugata inregistrarea (id=$id)";
 }
 elseif (isset($actualizeaza))
 {

```

```

$interogare = "UPDATE agenda_adrese SET nume='$nume', "
 "email='$email' WHERE id = $id";
$resultat = mysql_query($interogare);
check_mysql();
$mesaj = "A fost actualizata inregistrarea (id=$id)";
}
elseif (isset($sterge))
{
 $interogare = "DELETE FROM agenda_adrese WHERE id = $id";
 $resultat = mysql_query($interogare);
 check_mysql();
 $nume = "";
 $email = "";
 $mesaj = "A fost stersa inregistrarea (id=$id)";
}
$nume = trim($nume);
$email = trim($email);

?>

Nume:

<INPUT TYPE="TEXT" NAME="nume"
 <?php echo "VALUE=\"\$nume\" \" ?>>

Adresa de e-mail:

<INPUT TYPE="TEXT" NAME="e-mail"
 <?php echo "VALUE=\"\$e-mail\" \" ?>>

<INPUT TYPE="SUBMIT" NAME="stanga" VALUE="<">
<INPUT TYPE="SUBMIT" NAME="dreapta" VALUE=">">
<INPUT TYPE="SUBMIT" NAME="cauta" VALUE="Cauta">

<INPUT TYPE="SUBMIT" NAME="adauga" VALUE="Adauga">
<INPUT TYPE="SUBMIT" NAME="actualizeaza" VALUE="Actualizeaza">
<INPUT TYPE="SUBMIT" NAME="sterge" VALUE="Sterge">
<INPUT TYPE="HIDDEN" NAME="id"
 <?php echo "VALUE=\"\$id\" \" ?>>
<?php
if (isset($mesaj))
{
 echo "

$mesaj";
}
?>
</FORM>
</BODY>
</HTML>

```

5. Studiați scriptul, pentru a înțelege modul de funcționare a acestuia. Este util să comparați acest script cu scriptul similar asociat proiectului 11-3. Scriptul este destul de lung, dar o bună parte a programului este inclusă într-o instrucțiune if-elseif care constă din numeroase instrucțiuni compuse, incluse între paranteze acolade (fiecare instrucțiune compusă este de sine stătătoare și absolut inteligibilă ca atare). Deci, studiați-le una câte una.
6. În particular, observați că interogările efectuate la executarea unui clic pe butoanele de navigare (< și >) folosesc identificatorul de rând atribuit automat de MySQL. De asemenea, remarcați că rezultatul interogării este sortat, astfel încât rândul corespunzător –

rândul asociat intrării din agenda cu adrese plasată imediat înainte sau după intrarea curentă – este primul rând din setul de rezultate.

7. Orientați un browser Web spre adresa URL asociată scriptului. Fereastra browserului trebuie să fie similară celei prezentate alăturat:
8. Verificați modul de operare a scriptului. Acesta trebuie să permită deplasarea înainte (>) respectiv înapoi (<) în interiorul agendei cu adrese. Trebuie să puteți adăuga, actualiza, respectiv șterge intrările din agenda cu adrese. De asemenea, scriptul trebuie să permită căutarea primei intrări din agenda cu adrese care corespunde unui șir de căutare specificat.

### Test de evaluare

1. Scrieți un program PHP care se conectează la un server MySQL plasat la gazda numită db, folosind identificatorul de utilizator admin și parola secret.
2. Scrieți un program PHP care selectează baza de date numită inventar în vederea unui acces ulterior.
3. Scrieți un program PHP care execută interogarea stocată în variabila șir \$sql și stochează rezultatul în variabila \$rset.
4. Scrieți un program PHP care afișează numărul erorii asociate celei mai recente interogări MySQL.
5. Scrieți un program PHP care afișează valoarea primei coloane a rândului următor al setului de rezultate stocat în variabila \$rset.
6. Scrieți o buclă PHP care parcurge prin iterație rândurile unui set de rezultate, plasând fiecare rând în variabila \$rand. Bucla va fi configurată astfel încât să aibă un corp fără conținut.

# Modulul 15:

## Utilizarea claselor și a obiectelor

### Scopuri

- **Învățați sensul și utilitatea conceptului de orientare spre obiecte**
- **Învățați să definiți și să creați instanțe ale claselor**
- **Învățați să definiți și să utilizați proprietățile și metodele**
- **Învățați să lucrați cu tablouri de obiecte**

O abordare importantă a activității de programare, care a devenit populară în anii '90, este *orientarea spre obiecte*. Unele limbaje de programare, precum Java, sunt în mod intrinsec orientate spre obiecte. De fapt, nu puteți scrie un program Java fără a înțelege și fără a utiliza clasele, obiectele, proprietățile și metodele. Prin contrast, PHP folosește o abordare mai puțin dogmatică. PHP vă permite să creați și să folosiți clasele, obiectele, respectiv proprietățile și metodele asociate acestora. De asemenea, PHP vă lasă toată libertatea de a crea programe care nu sunt orientate spre obiecte, dacă preferați.

### O prezentare introductivă a conceptului de orientare spre obiecte

Orientarea spre obiecte a fost creată ca o modalitate de organizare a unor programe de simulare complexe. Cu toate acestea, progresul interfețelor grafice cu utilizatorul din perioada anilor '90 a dus la descoperirea că metodele orientate spre obiecte facilitează scrierea programelor interactive.

Principiul de bază al orientării spre obiecte îl reprezintă *încapsularea*. Un program care nu este orientat spre obiecte este organizat ca un set de funcții și un set de variabile globale utilizate de acele funcții. Orice funcție poate opera folosind orice variabilă globală, deci structura programului este oarecum haotică. Prin contrast, un program orientat spre obiecte combină funcții și variabile conexe într-o unitate, cunoscută sub numele de *clasă*. Un program orientat spre obiecte caracteristic este alcătuit din mai multe clase.

Încapsularea asigurată de clase este un concept similar pereților de compartimentare și porților etanșe folosite în construcțiile navale, care împart nava în unități distincte. Într-un program orientat spre obiecte, funcțiile asociate cu o clasă nu pot obține acces în mod arbitrar la variabile asociate unei alte clase. Programele orientate spre obiecte pot fi mai ușor de înțeles decât programele care nu sunt orientate spre obiecte, deoarece interacțiunile între clasele unui program orientat spre obiecte

sunt relativ reduse ca număr. Ca atare, un programator poate studia și înțelege un program orientat spre obiecte abordând fiecare clasă în parte, nu toate clasele deodată.

Secțiunea următoare prezintă conceptele orientării spre obiecte într-un mod general, fără o referire concretă la modalitatea în care PHP implementează orientarea spre obiecte. Ulterior pe parcursul acestui modul, veți învăța să folosiți caracteristicile orientate spre obiecte ale limbajului PHP.

### Clase

Dacă doriți, puteți asimila o clasă cu un tip definit de utilizator. Orientarea spre obiecte vă permite să extindeți tipurile PHP standard – întreg, cu virgulă mobilă și șir – pentru a include tipuri create de dumneavoastră, precum `ContBancar`, `Client`, `Scranciob` sau `SitWeb`. Când definiți o clasă, îi descrieți caracteristicile. Acestea includ:

- **Proprietăți** – variabile care descriu membrii clasei
- **Metode** – operații pe care membrii clasei le pot efectua

De exemplu, să presupunem că definiți o clasă care reprezintă un cont bancar. Clasa poate include proprietăți precum următoarele:

- Numărul contului
- Posesorul contului
- Data înființării contului
- Balanța curentă

Probabil că vor mai fi adăugate și alte proprietăți, precum informațiile de contact aferente posesorului contului.

Metodele unei clase sunt similare unor funcții care obțin acces la valorile și proprietățile unei clase, respectiv le modifică. Deseori, metodele sunt legate de evenimente și sunt responsabile cu modificarea valorilor proprietăților astfel încât să reflecte apariția unui eveniment. Clasa `ContBancar` poate include metode precum:

- Crearea unui depozit
- Efectuarea unei retrageri
- Obținerea balanței curente
- Închiderea contului

De asemenea, o clasă poate include o metodă specială, numită *metodă constructor* sau pur și simplu *constructor*. Constructorul unei clase este folosit pentru a crea instanțe sau membri ai clasei, care sunt cunoscuți sub numele de *obiecte*. Constructorul clasei `ContBancar` va crea un obiect `ContBancar`; această operație se va executa la fiecare deschidere a unui cont nou.

## Sugestie

Diferența între conceptul de clasă și cel de obiect este importantă. Dacă o clasă poate fi asimilată unui tip de date, un obiect poate fi echivalat cu o variabilă sau cu o valoare având un anumit tip de date. O clasă mai poate fi asemănată cu un șablon care este utilizat pentru a specifica și pentru a crea entități cunoscute sub numele de *obiecte*. Practic, o clasă este o „fabrică” de obiecte, care produce obiecte cu aceeași structură, adică obiecte având proprietăți și metode identice.

## Moștenire

Puterea conceptului de orientare spre obiecte se bazează în mare măsură pe o caracteristică numită *moștenire*. Aceasta vă permite să specificați o clasă folosind o altă clasă ca punct de plecare. Clasa originală se numește *clasă de bază* sau *clasă părinte*, clasa specificată mai recent se numește *clasă derivată* sau *clasă copil*.

De exemplu, clasa `ContBancar` descrisă în secțiunea anterioară se poate folosi pentru specificarea a două noi clase: `ContCurent` și `ContEconomii`. O clasă copil moștenește proprietățile și metodele părintelui său. În consecință, atât clasa `ContCurent`, cât și clasa `ContEconomii` vor avea proprietăți care reprezintă numărul contului, posesorul contului, data înființării contului și balanța curentă. De asemenea, cele două clase vor avea metode capabile de a înregistra depunerile și retragerile, precum și de a închide contul.

Principalul avantaj al moștenirii îl constituie economia. Puteți alege să definiți clasa `ContCurent` fără a face referire la clasa `ContBancar`. Prin definirea clasei `ContCurent` drept copil al clasei `ContBancar` se realizează, însă, o economie de efort.

În cazul în care `ContBancar` ar fi fost o clasă relativ simplă, economiile nu ar fi fost importante. Dar, în principiu, o clasă părinte poate conține zeci sau sute de proprietăți și metode, pe care o clasă copil le poate moșteni aproape fără efort.

Mai mult, moștenirea poate facilita întreținerea unui program. Să presupunem că o lege nouă impune băncilor să asocieze fiecărui cont un număr special. În cazul în care clasele `ContCurent` și `ContEconomii` ar fi fost definite pornind de la zero, atunci ar fi fost necesară revizuirea ambelor clase. Dar, în cazul în care clasele ar fi fost definite drept clase copil, ar fi fost necesară numai revizuirea clasei `ContBancar`; clasele copil ale acestora vor moșteni modificările aduse.

Un alt potențial avantaj al moștenirii constă în reutilizarea liniilor de program. Moștenirea permite utilizarea unei clase chiar dacă aceasta este *aproape* cea necesară pentru un scop dat, fără a fi exact clasa adecvată scopului respectiv. Așa cum se va explica ulterior în acest modul, metodele inadecvate ale clasei pot fi redefinite sau anulate de metodele dintr-o clasă copil. Această posibilitate de adaptare a unei clase astfel încât aceasta să corespundă unei diversități de contexte și aplicații permite reutilizarea unei clase mai frecvent decât ar fi fost posibil în caz contrar.



## Sfatul specialistului

**Întrebare:** Ați explicat noțiunea de *orientare spre obiecte*, dar eu am auzit și de expresia *bazat pe obiecte*. Ce înseamnă aceasta?

**Răspuns:** Un sistem sau limbaj bazat pe obiecte este unul care furnizează un set de clase predefinite. Aveți toată libertatea de a crea instanțe (obiecte) ale acestor clase. Cu toate acestea, nu aveți permisiunea de a specifica clase noi. Unii producători își descriu limbajele sau sistemele ca fiind *orientate* spre obiecte, chiar dacă acestea nu sunt decât *bazate* pe obiecte. Trebuie să investigați cu atenție afirmațiile acestora privind orientarea spre obiecte, pentru a vă asigura că este disponibilă întreaga gamă a facilităților furnizate de acest concept.



## Test „la minut”

- Principiul orientării spre obiecte care divide un program în clase la care sunt asociate proprietăți și metode corelate se numește \_\_\_\_\_.
- Un membru sau instanță al(a) unei clase se numește \_\_\_\_\_.
- Facilitatea care permite unei clase să dețină proprietățile și metodele unei alte clase se numește \_\_\_\_\_.
- O clasă folosită pentru a specifica o clasă nouă se numește \_\_\_\_\_.

## Definirea și instanțierea unei clase

Pentru a vedea care este modalitatea de definire a unei clase în PHP, să considerăm o clasă simplă, care reprezintă un cont bancar. Iată instrucțiunile PHP folosite pentru definirea acestei clase:

```
class ContBancar
{
 var $balanta = 0;

 function creeaza_depozit($suma)
 {
 $this->balanta = $this->balanta + $suma;
 }

 function obtine_balanta()
 {
 return $this->balanta;
 }
}
```

Răspunsuri la test:

- Încapsulare
- Obiect
- Moștenire
- Clasă părinte sau clasă de bază

Clasa are o singură proprietate, și anume \$balanta. De asemenea, clasa mai are două metode, în speță creeaza\_depozit() și obtine\_balanta().

Numele clasei este specificat de către instrucțiunea class. Definiția clasei este delimitată prin două paranteze acolade. Proprietatea \$balanta este definită de o instrucțiune var. În exteriorul claselor PHP, instrucțiunea var este rareori folosită, deoarece PHP definește în mod automat o variabilă în momentul atribuirii unei valori. Cu toate acestea, proprietățile claselor trebuie definite folosind instrucțiunea var. Clasa ContBancar atribuie proprietății \$balanta valoarea inițială zero. Nu este necesară atribuirea unei valori inițiale unei proprietăți, dar astfel programele devin mai simple și mai ușor de citit.

Observați că metodele sunt definite folosind cuvântul cheie function, ceea ce este normal, deoarece metodele, ca și funcțiile, execută acțiuni. Diferența dintre metode și funcții este legată de amplasarea acestora; metodele sunt definite în interiorul claselor, în timp ce funcțiile sunt definite în exteriorul acestora.

Cel mai interesant aspect al metodelor constă în modalitatea bizară în care acestea obțin accesul la \$nume. Sintaxa \$this-> are o semnificație similară cu aceea a pronumelui posesiv din limba română, persoana I singular (*meu*). Când o metodă face o referire de forma \$this->balanta, se consideră că referința face trimitere la proprietatea \$balanta a obiectului curent. PHP stabilește valoarea variabilei speciale \$this la fiecare invocare a unei metode, așa cum veți vedea în sub-secțiunea următoare. Efectul este similar cu acela al expresiei „balanța mea”.

## Instanțierea unui obiect

Pentru a determina modul de funcționare a clasei ContBancar, să examinăm câteva linii de program care instanțiază și folosesc un obiect ContBancar:

```
$cont = new ContBancar();
$cont->creeaza_depozit(100);
echo "Balanta este " . $cont->obtine_balanta();
```

Prima instrucțiune instanțiază un obiect ContBancar folosind operatorul new. Instrucțiunea stochează o referință la obiect în variabila \$cont, care va fi utilizată ulterior pentru a obține acces la metodele obiectului.

Cea de-a doua instrucțiune invocă metoda creeaza\_depozit() a obiectului, furnizând o valoare a argumentului egală cu 100. Dacă reveniți la definiția clasei, veți vedea că efectul acestei metode constă din incrementarea valorii proprietății \$balanta, operație efectuată, evident, de o metodă numită creeaza\_depozit().

Remarcați că a doua instrucțiune folosește variabila \$cont, care face referire la obiectul ContBancar creat de prima instrucțiune. Operatorul -> arată că PHP trebuie să invoce metoda creeaza\_depozit() asupra obiectului desemnat prin variabila \$cont. În timpul invocării metodei, obiectul curent este obiectul desemnat prin \$cont.

Deci, în timpul invocării metodei, variabila specială \$this face referire la același obiect ca și variabila \$cont.

Cea de-a treia instrucțiune invocă metoda obtine\_balanta(), care returnează valoarea proprietății \$balanta. Instrucțiunea afișează valoarea returnată prin intermediul unei instrucțiuni echo.

Observați că metoda obtine\_balanta() obține accesul la valoarea proprietății obiectului, dar nu modifică valoarea acesteia. Asemenea metode sunt frecvent folosite și se numesc *metode acces* sau *metode de obținere*. Metode precum creeaza\_depozit(), care modifică valoarea unei proprietăți, se numesc *metode mutator* sau *metode de configurare*.

## Definirea unei metode constructor

Din clasa ContBancar lipsesc o mulțime de proprietăți esențiale pe care trebuie să le conțină chiar și clasa cea mai simplă, de exemplu numele posesorului contului. Totuși, omisiunea a permis definirea unei clase fără constructor, deoarece nu a fost necesară nici o informație pentru a crea o instanță a clasei. Iată o versiune revizuită a clasei, care include o metodă constructor:

```
class ContBancar2
{
 var $cont_id;
 var $nume_posesor;
 var $balanta = 0;

 function ContBancar2($id, $nume, $suma)
 {
 $this->cont_id = $id;
 $this->nume_posesor = $nume;
 $this->balanta = $suma;
 }

 function creeaza_depozit($suma)
 {
 $this->balanta = $this->balanta + $suma;
 return $this->balanta;
 }
}
```

Constructorul este, așa cum se observă din exemplu, o funcție cu același nume ca al clasei. Diferența este că PHP apelează automat această funcție la invocarea operatorului new.

Iată cum s-ar putea folosi clasa revizuită:

```
$cont = new ContBancar2(1001, "Fane Filantropu", 1000);
echo "Contul $cont->cont_id apartine lui $cont->nume_posesor";
$cont->creeaza_depozit(100);
echo "
Balanta este " . $cont->balanta;
```



Argumentele constructorului sunt folosite pentru a stabili valorile inițiale ale proprietăților noului obiect.

Remarcați modul în care a doua și a patra instrucțiune obțin accesul la proprietățile obiectului în mod direct, nu prin intermediul metodelor accesoriu, așa cum se procedează în exemplul cu clasa ContBancar. Spre deosebire de majoritatea celorlaltor limbaje orientate spre obiecte, PHP nu împiedică acest tip de acces; ba chiar permite modificarea directă a valorilor proprietăților. Să examinăm următorul exemplu:

```
$cont->balanta = 1000;
echo "
Balanta este " . $cont->balanta;
```

Prima instrucțiune modifică valoarea proprietății \$balanta în mod direct, nu prin invocarea metodei creeaza\_depozit(). Programele care obțin accesul la proprietățile obiectelor și le modifică în mod direct distrug încapsularea obiectelor și implicit subminează potențialele avantaje ale orientării spre obiecte.



### Sfatul specialistului

**Întrebare:** Dacă distrugerea încapsulării nu este recomandată, de ce PHP permite programatorilor această operație?

**Răspuns:** Inițial, PHP nu a fost conceput ca un limbaj orientat spre obiecte. Caracteristicile orientate spre obiecte au fost adăugate ulterior. Deoarece dezvoltarea limbajului PHP continuă, este posibil ca, în cele din urmă, să includă mecanisme care să impună respectarea încapsulării. Lipsa unor asemenea mecanisme nu constituie un handicap major atunci când un individ sau o mică echipă colaborează pentru implementarea unui sistem de mici dimensiuni, dar asemenea mecanisme sunt esențiale atunci când echipe de dezvoltatori abordează sisteme de anvergură. În caz contrar, numărul defectelor software riscă să devină inacceptabil de mare.



### Test „la minut”

- Care este cuvântul cheie folosit la definirea proprietății unui obiect în PHP?
- Care este cuvântul cheie folosit la definirea metodei unui obiect în PHP?
- Cum este posibilă identificarea constructorului unei clase numite Test?

Răspunsuri la test:

- var
- function
- Este o funcție numită Test

## Utilizarea moștenirii

Așa cum s-a explicat anterior, o bună parte din avantajele utilizării conceptului de orientare spre obiecte este asociat cu facilitatea cunoscută sub numele de moștenire. Iată un exemplu care prezintă modul de derivare a unei clase din clasa ContBancar2, definită anterior:

```
include "contbancar2.php";
class ContCurent extends ContBancar2
{
 var $cec_nr;

 function ContCurent($id, $nume, $suma, $cecnr)
 {
 $this->cont_id = $id;
 $this->nume_posesor = $nume;
 $this->balanta = $suma;
 $this->cec_nr = $cecnr;
 }

 function incasare_cec($cecnr, $suma)
 {
 $this->balanta = $this->balanta - $suma;
 }

 function obtine_balanta()
 {
 return $this->balanta;
 }
}
```

Exemplul folosește o instrucțiune include pentru a obține accesul la definiția clasei ContBancar2; totuși, este de asemenea posibil ca definițiile clasei părinte și ale clasei copil să fie plasate în același fișier. Cuvântul cheie extends apare în instrucțiunea class și stabilește identitatea clasei ContCurent drept copil al clasei ContBancar2. Clasa copil definește o proprietate, \$cec\_nr, care conține numărul filei inițiale a următorului carnet de cecuri care va fi emis; de asemenea, definește un constructor și două metode, în speță incasare\_cec() și obtine\_balanta(). Cel mai important este că această clasă mai include proprietățile \$cont\_id, \$nume\_posesor și \$balanta, precum și o metodă creeaza\_depozit(), care sunt definite în clasa părinte. Constructorul atribuie valori proprietăților moștenite, dar și proprietății definite în clasa copil.

Iată instrucțiunile care creează un obiect ContCurent și care invocă numeroase metode asupra acestuia:

```
$cont = new ContCurent(101, "Misu Mizerie", 100, 1101);
$cont->incasare_cec(1001, 250);
$cont->creeaza_depozit(350);
echo "Balanta curenta: " . $cont->obține_balanta;
```



Observați că metoda `creaza_depozit()`, care este definită în clasa părinte, este invocată ca și cum ar fi fost definită în clasa copil; metodele moștenite nu necesită un tratament de natură specială.

## Redefinirea metodelor

Uneori, o clasă existentă include o metodă care nu este adecvată pentru o clasă care în alte condiții poate fi derivată pentru a obține o clasă copil. În loc de a defini noua clasă fără referire la clasa existentă, puteți redefini metoda inadecvată. De exemplu, să considerăm următoarea clasă:

```
class ContBancar3
{
 var $cont_id;
 var $nume_posesor;
 var $balanta = 0;

 function ContBancar3($id, $nume, $suma)
 {
 $this->cont_id = $id;
 $this->nume_posesor = $nume;
 $this->balanta = $suma;
 }

 function inchide_cont()
 {
 $suma = $this->balanta;
 $this->balanta = 0;
 return $suma;
 }
}
```

Să presupunem că doriți să derivați o clasă care reprezintă un nou tip de cont bancar purtător de dobândă. La închiderea contului, programul trebuie să calculeze dobânda acumulată de la ultima declarație lunară și să returneze clientului atât dobânda, cât și balanța curentă.

Iată cum puteți proceda:

```
class ContEconomii extends ContBancar3
{
 function inchide_cont($zile, $rata)
 {
 $suma = $this->balanta * $rata * ($zile / 360);
 $suma = $suma + $this->balanta;
 $this->balanta = 0;
 return $suma;
 }
}
```

Remarcați că în cadrul clasei copil este definită o metodă numită `inchide_cont()`. O metodă cu același nume există și în clasa părinte, deși are o altă definiție.

Să presupunem că ați creat un obiect `ContEconomii` și că invocați metoda `inchide_cont()` astfel:

```
$cont = new ContEconomii(101, "Zozo Zgarcitul", 1000);
echo "Zozo primește " . $cont->inchide_cont(18, 0.05);
```

Este invocată metoda definită de clasa copil, nu cea definită în clasa părinte. Se spune că metoda definită în clasa părinte a fost *anulată* (*redefinită*) de către metoda definită în clasa copil.

## Invocarea unei metode redefinite

Dacă încercați să invocați o metodă redefinită, probabil că veți obține o eroare. De exemplu, să presupunem că ați determinat PHP să execute următoarele instrucțiuni:

```
$cont = new ContEconomii(101, "Zozo Zgarcitul", 1000);
echo "Zozo primește " . $cont->inchide_cont();
```

Metoda `inchide_cont()` este redefinită în cadrul clasei `ContEconomii` de către o nouă metodă, care preia două argumente. Această încercare de a invoca metoda redefinită are ca rezultat mesaje de eroare:

```
Warning: Missing argument 1 for inchide_cont()
Warning: Missing argument 2 for inchide_cont()
(Avertisment: argumentele 1 și 2 pentru funcția inchide_cont() lipsesc.)
```

Dacă, în schimb, ați fi încercat să invocați metoda redefinită din interiorul clasei copil, ați fi obținut un rezultat ușor diferit, dar la fel de nedorit. De exemplu, să presupunem că rescrieți clasa copil astfel:

```
class ContEconomii extends ContBancar3
{
 function inchide_cont($zile, $rata)
 {
 $suma = $this->inchide_cont();
 $suma = $suma + $suma * $rata * ($zile / 360);
 return $suma;
 }
}
```

Prin executarea acestui script se obține mesajul de eroare

```
Warning: Missing argument 1 for inchide_cont()
```

Mai rău este că mesajul de eroare se repetă de mai multe ori. Metoda `inchide_cont()` se auto-apellează în mod repetat, până când serverul PHP ajunge la capătul răbdărilor și anulează cererea.

Cu toate acestea, este posibilă invocarea metodei redefinite. Nu este nevoie decât de un mic artificiu sintactic:

```
class ContEconomii extends ContBancar3
{
 function inchide_cont($zile, $rata)
 {
 $suma = ContBancar3::inchide_cont();
 $suma = $suma + $suma * $rata * ($zile / 360);
 return $suma;
 }
}
```

Prin specificarea numelui clasei, urmat de o pereche de caractere două puncte, puteți indica programului PHP să invoce metoda definită în clasa părinte, nu metoda definită în clasa copil.



### Sfatul specialistului

**Întrebare:** Să presupunem că se folosește clasa A pentru derivarea unei noi clase B. Apoi, clasa B este folosită pentru a deriva o altă clasă C, o clasă nepot a clasei A. Pot metodele din C să redefinească metodele din A și B? Este posibil ca metodele redefinite din A să fie apelate în B și C?

**Răspuns:** Da. Relația dintre o clasă bunic și o clasă nepot este similară cu aceea dintre o clasă părinte și o clasă copil. Clasa copil poate redefini metodele clasei sale părinte, respectiv bunic. De asemenea, poate face referire la metodele redefinite prin prefixarea numelui metodei redefinite cu numele clasei în care aceasta a fost inițial definită, urmat de o pereche de caractere două puncte.



### Test „la minut”

- Cum se numește o metodă definită atât într-o clasă părinte, cât și într-o clasă copil?
- Să presupunem că o clasă Parinte este folosită pentru a deriva clasa Copil, precum și că ambele clase definesc metoda test(). În cadrul clasei copil, cum puteți invoca metoda test() definită în clasa părinte?

## Tablouri cu obiecte

În cadrul modulelor anterioare, s-a explicat că tablourile reprezintă o modalitate convenabilă pentru lucrul cu mai multe valori. Valorile stocate în tablouri pot face referire la obiecte exact așa cum procedează în cazul întregilor, al valorilor cu virgulă mobilă sau al șirurilor. Un tablou care face referire la obiecte se numește *tablou cu obiecte*.

Răspunsuri la test:

- Redefinită
- Parinte::test()

Pentru a demonstra modul de utilizare a unui tablou cu obiecte, să examinăm următoarea clasă și cele două clase copil ale acesteia:

```
class ContBancar
{
 var $cont_id;
 var $nume_posesor;
 var $balanta;
 var $tip_cont;

 function ContBancar($id, $nume, $suma, $tip)
 {
 $this->cont_id = $id;
 $this->nume_posesor = $nume;
 $this->balanta = $suma;
 $this->tip_cont = $tip;
 }

 function dump()
 {
 return "Cont " . $this->cont_id
 . " posesor " . $this->nume_posesor
 . " balanta " . $this->balanta
 . " tip " . $this->tip_cont;
 }
}

class ContCurent extends ContBancar
{
 function ContCurent($id, $nume, $suma)
 {
 ContBancar::ContBancar($id, $nume, $suma, "curent");
 }
}

class ContEconomii extends ContBancar
{
 function ContEconomii($id, $nume, $suma)
 {
 ContBancar::ContBancar($id, $nume, $suma, "economii");
 }
}
```

Remarcați faptul că funcțiile constructor ale claselor copil fac referire la constructorul clasei părinte, reutilizând în mod eficient caracteristica furnizată de clasa părinte.

Acum, să presupunem că am creat instanțe ale claselor copil și că stocăm referințele la aceste instanțe într-un tablou:

```
$acct[0] = new ContCurent(101, "P. Pene", 1000);
$acct[1] = new ContEconomii(102, "C. Apa", 1200);
```

În exemplu sunt menționate numai două instanțe, dar într-o utilizare efectivă a procedurii cu tabloul de obiecte, într-un tablou pot fi stocate sute sau mii de

asemenea referințe. O dată referințele stocate în tablou, pot fi prelucrate în mai multe moduri. De exemplu, să luăm următorul program:

```
foreach ($cont as $contul) {
 echo "
" . $contul->dump();
}
```

Din moment ce referințele la obiecte incluse în tablou se referă întotdeauna la un obiect la care este asociată o metodă `dump()`, este o joacă de copil să se parcurgă tabloul prin iterație și să se invoce metoda `dump()`, care afișează proprietățile unui obiect `ContBancar` sau ale unui obiect dintr-una din clasele sale copil, precum `ContBancar` sau `ContEconomii`.

## ? Sfatul specialistului

**Întrebare:** Când folosesc metoda cu tabloul de obiecte, primesc mereu mesajul de eroare „Call to undefined function“ (apel la funcție nedefinită). De ce se întâmplă aceasta?

**Răspuns:** Mesajul arată că programul dumneavoastră a încercat să invoce o metodă (adică o funcție) asupra unui obiect a cărui clasă nu definește sau moștenește metoda.

Când stocați într-un tablou referințe la obiecte și apoi invocați o metodă asupra referințelor, este de datoria dumneavoastră să vă asigurați că metoda este definită sau moștenită de fiecare clasă conexă. Cea mai simplă modalitate de a proceda astfel este de a deriva fiecare clasă dintr-o singură clasă părinte, care definește fiecare metodă pe care o invocați. Astfel, prin moștenire sunteți asigurați că metoda este disponibilă.



### Test „la minut“

- În procedul cu tabloul de obiecte, un tablou stochează \_\_\_\_\_ la obiecte.
- Fiecare obiect stocat trebuie să fie membru al unei clase care \_\_\_\_\_ sau \_\_\_\_\_ fiecare metodă invocată.



### Proiect 15-1: Lucrul cu obiecte

În cadrul acestui proiect, veți folosi un obiect pentru încapsularea informațiilor despre stilurile de text folosite în paginile unui sit Web. Veți vedea modalitatea de păstrare a unui aspect consecvent în toate paginile sitului Web, precum și posibilitatea de a revizui aspectul prin modificarea unui

Răspunsuri la test:

- Referințe
- Definește, moștenește

singur fișier, nu a fiecărei pagini. Puteți efectua operații similare folosind foile de stil HTML. Totuși, suportul pentru foile de stil nu este încă fiabil și consecvent, deci este posibil ca abordarea prezentată în acest proiect să corespundă într-un mod mai adecvat necesităților dumneavoastră.

### Scopurile proiectului

- Prezentarea modului de creare și utilizare a obiectelor într-un program mic, dar complet
- Prezentarea modului de încapsulare a informațiilor despre stil în cadrul unui obiect PHP

### Pas cu pas

1. Creați următorul script PHP, denumiți-l `stdpage.inc` și încărcăți-l în serverul dumneavoastră PHP:

```
<?php
class StdStilPagina
{
 var $litera_antet = "Tahoma";
 var $marime_antet = "7";
 var $litera_corp = "Times New Roman";
 var $marime_corp = "5";
 var $litera_subsol = "Letter Gothic";
 var $marime_subsol = "2";

 function set_antet($litera, $marime)
 {
 $this->litera_antet = $litera;
 $this->marime_antet = $marime;
 }

 function set_corp($litera, $marime)
 {
 $this->litera_corp = $litera;
 $this->marime_corp = $marime;
 }

 function set_subsol($litera, $marime)
 {
 $this->litera_subsol = $litera;
 $this->marime_subsol = $marime;
 }

 function antet_text($text)
 {
 echo '<tip litera="' . $this->litera_antet . " marime=" .
 $this->marime_antet . ">';
 echo $text;
 echo "";
 }

 function subsol_text($text)
 {
 echo '<tip litera="' . $this->litera_subsol . " marime=" .
 $this->marime_subsol . ">';
 echo $text;
 echo "";
 }
}
```

```

}

function corp_text($text)
{
 echo "<tip litera=" . $this->litera_corp . " marime=" .
 $this->marime_corp . ">";
 echo $text;
 echo "";
}
}
?>

```

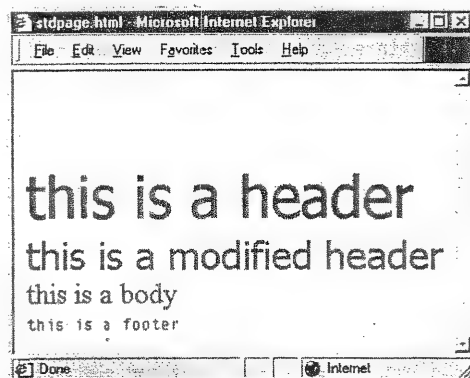
2. Studiați scriptul, observând proprietățile care definesc stilurile HTML pentru antetul, corpul și subsolul unei pagini Web. Remarcați metodele mutator, care vă permit să revizuiți stilurile. Observați și metodele accesori, care generează linii de program HTML care specifică atributele stilului antetului, al corpului sau al subsolului.
3. Creați următorul script, denumiți-l stdpage.php și încărcați-l în catalogul care conține scriptul stdpage.inc:

```

<HTML>
<HEAD>
<TITLE>stdpage.php</TITLE>
</HEAD>
<BODY>
<?php
require "stdpage.inc";
$stil = new StdStilPagina();
$stil->antet_text("
Acesta este un antet");
$stil->set_antet("Tahoma", 6);
$stil->antet_text("
Acesta este un antet modificat");
$stil->corp_text("
Acesta este un corp");
$stil->subsol_text("
Acesta este un subsol");
?>
</BODY>
</HTML>

```

4. Studiați scriptul, care este o pagină Web model ce folosește clasa definită în scriptul stdpage.inc, accesibil prin intermediul instrucțiunii require. Observați modul de instanțiere al obiectului StdStilPagina, precum și modul în care sunt utilizate metodele sale pentru a genera textul aferent antetului, corpului și subsolului.
5. Orientați browserul dumneavoastră Web spre adresa URL asociată scriptului stdpage.php. Rezultatul trebuie să fie asemănător celui prezentat în figura alăturată:
6. În cazul în care calculatorul dumneavoastră nu este configurat astfel încât să accepte corpurile de literă specificate în script, aspectul datelor dumneavoastră de ieșire poate varia. În acest caz, revizuiți scriptul astfel încât să facă referire la corpuri de literă disponibile. Puteți chiar specifica o listă de corpuri de literă, unde fiecare corp de literă este separat de vecinul său prin intermediul unei virgule.



7. Gândiți-vă că, folosind un procedeu similar celui prezentat aici, puteți obține o consecvență a stilurilor pentru un întreg sit Web. Mai mult, prin modificarea conținutului paginii stdpage.inc puteți revizui cu ușurință, în orice moment, stilurile folosite pentru antet, corp și subsol. În plus, puteți generaliza cu ușurință acest procedeu astfel încât acesta să includă atribute diferite de cele ale etichetei FONT și ale construcțiilor HTML (în afara textelor). În modulul 16 este prezentată o altă modalitate de satisfacere a dezideratelor legate de consecvență și mentenabilitate.



### Test de evaluare

1. Care este operatorul PHP folosit pentru instanțierea unui obiect?
2. Care este cuvântul cheie folosit pentru definirea unei clase?
3. Care este denumirea corectă a variabilelor incluse în cadrul unei clase?
4. Care este denumirea corectă a funcțiilor incluse în cadrul unei clase?
5. Care este denumirea corectă a funcției speciale folosite la crearea unui obiect?
6. Cum se mai numește o clasă părinte?
7. Cum se mai numește o clasă copil?
8. Cum se numește o metodă care este redefinită de o clasă copil?
9. Cum se numește o metodă care obține acces la valoarea unei proprietăți, dar nu o modifică?
10. Cum se numește o metodă care modifică valoarea unei proprietăți?

# Modulul 16:

## Utilizarea șabloanelor de aplicație

### Scopuri

- Învățați care este utilitatea șabloanelor
- Învățați să creați un șablon folosind FastTemplate
- Învățați să generați o pagină Web pornind de la un șablon
- Învățați să aplicați șabloanele pe tot cuprinsul unui sit Web

PHP este ideal pentru crearea de pagini Web și mici situri Web. Totuși, când folosiți PHP pentru implementarea de situri Web de mari dimensiuni, apar unele probleme. De exemplu, este dificilă păstrarea unei consecvențe a structurii unei pagini Web în toate paginile unui sit mare. Mai mult, colaborarea între graficieni și programatorii PHP poate fi dificilă, deoarece fiecare are cunoștințe destul de reduse despre activitatea celuilalt.

Clasele de șabloane sunt concepute pentru a remedia aceste probleme, precum și alte probleme legate de dezvoltarea siturilor de mari dimensiuni. În cadrul acestui modul, veți învăța despre clasa FastTemplate, o modalitate de utilizare a șabloanelor PHP.

## O prezentare introductivă a șabloanelor

Când un proiect de dezvoltare a unui sit Web capătă o amploare suficient de mare astfel încât să implice numeroși dezvoltatori, gestionarea proiectului devine mai complexă. Acest lucru este adevărat mai ales dacă dezvoltatorii sunt foarte specializați într-un anumit domeniu; de exemplu, dacă echipa de dezvoltare include dezvoltatori PHP care nu au cunoștințe în domeniul graficii, respectiv graficieni care nu au noțiuni de PHP. În asemenea situații, problemele pot apărea când graficienii sunt incapabili de a folosi editoarele HTML de tip WYSIWYG, precum Macromedia Dreamweaver sau Adobe GoLive, datorită liniilor de program PHP înglobate în paginile HTML. Graficienii nu pot întreține programele PHP, iar programatorii PHP nu pot manipula elementele de grafică. Ca atare, toată munca trebuie „pasată” încoace și încolo, cu stângăcie, de la un grup la altul și viceversa. Rezultatul? Ineficiență și eroare.

Prin contrast, șabloanele permite separarea programelor PHP și HTML în fișiere distincte, astfel încât programatorii PHP să poată lucra cu programe PHP, în timp ce graficienii să lucreze la programele HTML. Se poate beneficia de avantajele specializării, deoarece graficienii nu trebuie să interacționeze niciodată cu programele PHP.

În realitate, șabloanele sunt instrumente foarte simple. Iată un mic exemplu:

```
<HTML>
<HEAD>
<TITLE>Un exemplu de șablon</TITLE>
</HEAD>
<BODY>
Raspunsul la intrebarea de astazi este {RASPUNS}.
</BODY>
</HTML>
```

După cum puteți vedea, șablonul constă din linii de program HTML obișnuite, pe care un grafician le poate înțelege și pe care un editor HTML le acceptă. „Magia” stă în variabila șablon, și anume {RASPUNS}. Când este preluată pagina Web asociată șablonului, un script PHP determină valoarea variabilei {RASPUNS} și înlocuiește variabila cu valoarea acesteia.

Scriptul PHP este complet indiferent față de elementele HTML din pagină, elemente pe care, în consecință, un grafician are toată libertatea de a le modifica, iar graficianul nu este interesat de rezultatul care înlocuiește variabila (cu condiția ca toate liniile HTML care apar în textul înlocuitor să fie compatibile cu liniile de program HTML înconjurătoare, de care este responsabil graficianul). Atât programatorul PHP, cât și graficianul se bucură, ca atare, de libertatea relativă de a-și desfășura activitatea în mod independent. Graficianul are control asupra liniilor de program HTML și implicit asupra aspectului paginilor Web, în timp ce programatorul PHP deține controlul asupra elementelor de prelucrare a datelor care constituie suportul paginilor Web.

FastTemplate este una din numeroase modalități de susținere a șabloanelor în PHP, care mai includ PHP Base Library ([phplib.net/use.de](http://phplib.net/use.de)) și Xtemplate ([phpclasses.upperdesign.com](http://phpclasses.upperdesign.com)). Un avantaj cheie al procedurii FastTemplate este acela că nu impune o integrare cu serverul PHP. Deci, chiar dacă nu sunteți administratorul serverului PHP pe care îl folosiți, puteți descărca, instala și folosi FastTemplate.

La origine, FastTemplate a fost modulul Perl CGI::FastTemplate, scris de Jason Moore. Modulul Perl a fost portat în PHP de Joe Harris și este disponibil la adresa <http://www.thewebmasters.net>. FastTemplate poate fi redistribuit în condițiile licenței GNU General Artistic License, cu specificații ulterioare formulate de Jason Moore și Joe Harris.

## Utilizarea unui șablon

Pentru a folosi un șablon, trebuie să efectuați următoarele operații:

- Creați fișierul șablon
- Creați un script PHP care completează șablonul, prin executarea următoarelor operații:
  1. Instanțiați un obiect FastTemplate.

2. Asociați o variabilă șablon cu fișierul șablon.
3. Atribuiți valori variabilelor șablon.
4. Separați variabila șablon asociată fișierului șablon.
5. Afișați valoarea variabilei șablon care conține rezultatul.

În secțiunea de față vom detalia operațiile prezentate anterior.

## Crearea fișierului șablon

Prima etapă în utilizarea unui șablon constă în crearea unui fișier șablon. Un fișier șablon este similar cu un fișier HTML sau PHP obișnuit, cu două excepții:

- Prin convenție, numele fișierului are extensia .tpl
- Fișierul conține variabile șablon, care sunt identificate prin paranteze acolade de delimitare {}

Iată un exemplu simplu de fișier șablon:

```
<HTML>
<HEAD>
<TITLE>{TITLU}</TITLE>
</HEAD>
<BODY>
<H1>Bine ati venit la {SIT}</H1>
Ora locala este acum {ORA}.
</BODY>
</HTML>
```

Fișierul conține trei variabile șablon, și anume {TITLU}, {SIT} și {ORA}. Un script PHP va folosi clasa FastTemplate pentru a asocia valori acestor variabile și pentru a returna rezultatul unui browser Web.

## Instanțierea unui obiect FastTemplate

Următoarea etapă în utilizarea șabloanelor constă în crearea unui script PHP care va completa șablonul. Prima operație pe care trebuie să o execute scriptul PHP constă din instanțierea unui obiect FastTemplate:

```
include "class.FastTemplate.php";
$tpl = new FastTemplate('.');
```

Programul PHP care intră în alcătuirea clasei FastTemplate se găsește în fișierul class.FastTemplate. Instrucțiunea include pune clasa la dispoziția scriptului PHP. Argumentul funcției constructor precizează catalogul care conține fișierele șablon. Șirul '.' arată că șabloanele sunt rezidente în același catalog ca și scriptul PHP.

## Asocierea unei variabile șablon cu un fișier șablon

În continuare, scriptul PHP trebuie să asocieze o variabilă șablon cu fișierul șablon. Vom presupune că fișierul șablon se numește baza.tpl și că variabila șablon baza urmează a fi asociată cu fișierul șablon. Următoarele instrucțiuni PHP asociază variabila cu fișierul șablon:

```
$tpl->define(array('home' => 'baza.tpl'));
```

Rețineți că o variabilă șablon nu este decât un șir text, nu o variabilă PHP. Variabilele de tip șablon nu sunt incluse în expresii sau argumente; acestea sunt transferate metodelor FastTemplate și sunt returnate de către acestea.

În cadrul unei aplicații complexe, puteți fi în situația de a lucra cu mai multe fișiere șablon simultan. Metoda define() a clasei FastTemplate acceptă mai multe perechi variabilă – nume de fișier. De exemplu, următoarea instrucțiune asociază două variabile de tip șablon cu nume de fișiere:

```
$tpl->define(array('baza' => 'baza.tpl',
 'navbar' => 'navbar.tpl'));
```

## Atribuirea de valori variabilelor de tip șablon

Pentru a atribui o valoare unei variabile de tip șablon, invocați metoda assign(). De exemplu, iată trei instrucțiuni de atribuire referitoare la variabile de tip șablon specificate în fișierul șablon menționat anterior:

```
$tpl->assign('TITLU', 'FastTemplate Demo');
$tpl->assign('SIT', 'www.test.com');
$tpl->assign('ORA', date("h:i:s A"));
```

De exemplu, prima instrucțiune de atribuire asociază valoarea 'FastTemplate Demo' cu variabila șablon TITLU. Variabila respectivă este menționată sub forma {TITLU} în fișierul șablon.

## Separarea variabilei șablon asociate fișierului șablon

Pentru a înlocui referințele la variabilele șablon cu valorile acestora, invocați metoda parse(). De exemplu, instrucțiunea

```
$tpl->parse('REZULTAT', 'baza');
```

separă variabila șablon baza, care este asociată cu fișierul șablon baza.tpl. Toate referințele la variabilele șablon pentru care sunt definite valori sunt înlocuite cu valorile corespunzătoare. Rezultatul este atribuit variabilei șablon REZULTAT.



## Afișarea variabilei șablon care conține rezultatul

Clasa `FastTemplate` include o metodă, numită `FastPrint()`, care afișează valoarea unei variabile șablon. Prin invocarea metodei respective asupra variabilei șablon care conține rezultatul analizei fișierului șablon determină clasa `FastTemplate` să emită liniile de program HTML dorite:

```
$tpl->FastPrint('REZULTAT');
```

## Construcția unui sit Web complet

O pagină Web caracteristică include elemente precum următoarele:

- **Antete.** Apar în partea de sus a paginii.
- **Bară de navigare.** În general, apare sub antete sau pe latura din stânga a paginii.
- **Conținut.** Apare în partea centrală a paginii Web; de asemenea, mai poate apărea pe latura din stânga, respectiv din dreapta a paginii.
- **Subsoluri.** Apar în partea de jos a paginii.

Pentru a păstra consecvența acestor elemente pe întreg cuprinsul unui sit Web, în general este convenabil să se creeze un fișier șablon care descrie conținutul și aspectul fiecărui element. Un alt fișier șablon descrie modalitatea de combinare a elementelor astfel încât să formeze fiecare pagină. Proiectul 16-1 prezintă modul de aplicare a acestei tehnici la scară mică, ilustrând principiile care pot fi aplicate și siturilor Web de mari dimensiuni.

## ? Sfatul specialistului

**Întrebare:** Am descărcat clasa `FastTemplate`, dar primesc mesaje de eroare PHP atunci când încerc să o folosesc. Ce s-a întâmplat?

**Răspuns:** Mai întâi, asigurați-vă că ați plasat fișierul descărcat care conține clasa `FastTemplate` în catalogul corespunzător. Apoi, asigurați-vă că fișierul are extensia adecvată pentru serverul dumneavoastră PHP; în general, această extensie este `.php`.

În cazul în care clasa se încorporează să nu funcționeze, este necesar să o modificați. Clasa `FastTemplate` a fost proiectată pentru a funcționa cu PHP3. Sintaxa folosită în PHP4 este ușor diferită de sintaxa din PHP3; unele diferențe sunt semnificative și pot determina eșuarea clasei `FastTemplate`. Este de presupus că `FastTemplate` va fi în curând actualizată astfel încât să funcționeze cu PHP 4. Totuși, între timp, dacă descărcați versiunea 1.1 a clasei `FastTemplate`, lansată la 27 iunie 1999, modificați următoarele linii:

Linie	Conținut
174	<code>if (ereg("([A-Z0-9_+])",\$Line, \$unknown))</code>
199	<code>\$template = ereg_replace("\${key}","\$val","\$template");</code>
213	<code>if (ereg("([A-Z0-9_+])",\$template))</code>
373	<code>if( (!\$this-&gt;\$ParentTag) or (empty(\$this-&gt;\$ParentTag)) )</code>
413	<code>\$newParent .= "{\$MacroName}\n";</code>

astfel încât să aibă următorul conținut:

Linie	Conținut
174	<code>if (ereg("\{[A-Z0-9_+\}\}]", \$Line, \$unknown))</code>
199	<code>\$template = ereg_replace("\{\{key\}\}", "\$val", "\$template");</code>
213	<code>if (ereg("\{[A-Z0-9_+\}\}]", \$template))</code>
373	<code>if( (isset(\$this-&gt;\$ParentTag))</code> <code>or (!\$this-&gt;\$ParentTag) or (!\$this-&gt;\$ParentTag)</code> <code>or (empty(\$this-&gt;\$ParentTag)) )</code>
413	<code>\$newParent .= '{'."\$MacroName"}\n";</code>



### Test „la minut”

- Care este notația folosită pentru reprezentarea variabilelor șablon ale clasei `FastTemplate` într-un fișier șablon?
- Care este semnificația argumentului transferat metodei constructor a clasei `FastTemplate`?
- Care este metoda `FastTemplate` folosită pentru substituirea referințelor la variabilele șablon cu valorile corespunzătoare ale acestora?



## Proiect 16-1: Lucrul cu șabloane

În cadrul acestui proiect, veți folosi clasa `FastTemplate` pentru a crea o pagină Web generică, pagină ce poate servi drept șablon pentru un întreg sit Web.

### Scopurile proiectului

- Prezintă modul de creare și utilizare a șabloanelor `FastTemplate`
- Prezintă modul de încapsulare a elementelor recurente de proiectare a paginilor Web, astfel încât acestea să fie consecvente pe întreg cuprinsul sitului Web

### Pas cu pas

1. Creați următorul fișier șablon, denumiți-l `std.tpl` și încărcați-l în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>{titlu}</TITLE>
```

Răspunsuri la test:

- Variabilele șablon sunt delimitate prin paranteze acolade `{}`.
- Precizează catalogul în care rezidă fișierele șablon.
- `parse()`.

```

</HEAD>
<BODY>
<TABLE CELLPADDING="10">

<TR>
<TD COLSPAN="2" ALIGN="LEFT">{antet}</TD>
</TR>

<TR>
<TD VALIGN="TOP" ALIGN="LEFT">{stgnav}</TD>
<TD VALIGN="TOP" ALIGN="LEFT">{continut}</TD>
</TR>

</TABLE>
</BODY>
</HTML>

```

2. Studiați scriptul pentru a discerne între elementele de proiectare. Trebuie să descoperiți următoarele tipuri:

- Titlul paginii – reprezentat de variabila șablon {titlu}
- Antetul paginii – reprezentat de variabila șablon {antet}
- Bară de navigare – reprezentată de variabila șablon {stgnav}
- Conținutul paginii – reprezentat de variabila șablon {continut}

3. Creați următorul fișier șablon, atribuiți-i numele antet.tpl și încărcați-l în serverul dumneavoastră PHP:

```

<TABLE CELLPADDING="5" BORDER="1" WIDTH="600">
<TR>
<TD>
Situl Web generic

</TR>
</TABLE>

```

Șablonul conține conținutul care va apărea în partea de sus a fiecărei pagini Web a sitului. Un sit Web autentic va include, probabil, și elemente de grafică în antetul paginii. Totuși, acest exemplu include numai text, din motive de simplitate în utilizare.

4. Creați următorul fișier șablon, atribuiți-i numele stgnav.tpl și încărcați-l în serverul dumneavoastră PHP:

```

<TABLE CELLPADDING="5" BORDER="1" WIDTH="100">
<TR>
<TD>

Baza

Funcție 1

Funcție 2

Iesire

</TD>
</TR>
</TABLE>

```

Acest model conține legăturile și textul asociat care vor forma bara de navigare. Din nou, un sit Web autentic va include, probabil, un program JavaScript care creează butoane de derulare. Totuși, exemplul de față include numai text, din motive de claritate și simplitate în utilizare.

5. Creați următorul fișier șablon, atribuiți-i numele index.tpl și încărcați-l în serverul dumneavoastră PHP:

```

Bine ati venit la
{titlu}


```

Aici nu veti gasi informatii despre absolut nimic. Suntem atat de siguri ca nu veti gasi nimic interesant aici, incat suntem gata sa va platim daca ne demonstrati ca ne inselam.

Acest fișier include conținutul care va apărea în pagina de bază a sitului Web. Un sit Web autentic va include, probabil, și un conținut dinamic, care va atrage vizitatorii. Remarcați că textul liniei de titlu conține o referință la variabila șablon titlu. Aceeași variabilă a fost menționată și în fișierul șablon std.tpl.

6. Creați următorul script PHP, denumiți-l index.php și încărcați-l în serverul dumneavoastră PHP:

```

<?php
include 'class.FastTemplate.php';

$tpl = new FastTemplate('.');

$tpl->define(array('std' => 'std.tpl',
 'antet' => 'antet.tpl',
 'stgnav' => 'stgnav.tpl',
 'continut' => 'index.tpl'));

$tpl->assign('titlu', 'Situl Web generic');

$tpl->parse('antet', 'antet');
$tpl->parse('stgnav', 'stgnav');
$tpl->parse('continut', 'continut');
$tpl->parse('DUMMY', 'std');

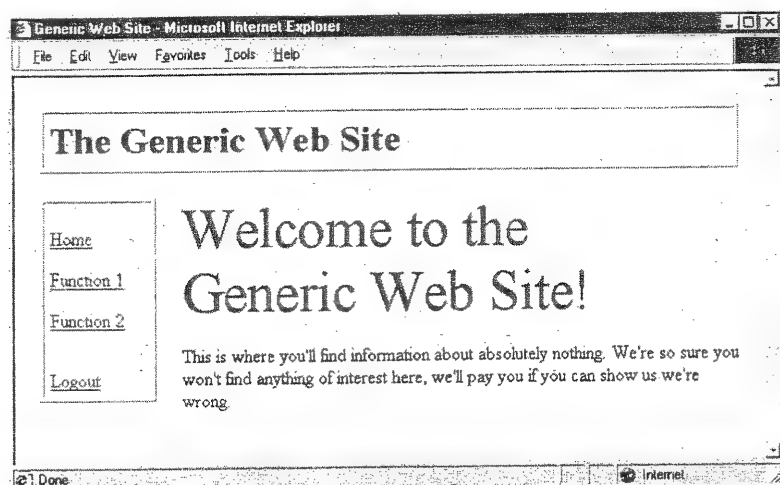
$tpl->FastPrint('DUMMY');

?>

```

Acest script PHP execută operația de completare a șabloanelor și urmărește îndeaproape șablonul descris anterior în cadrul acestui modul.

7. Orientați browserul dumneavoastră Web spre adresa URL asociată scriptului PHP `incex.php`. Ecranul browserului trebuie să fie asemănător cu următoarea ilustrație:



Structura defalcată a paginii nu este extrem de interesantă și nici măcar plăcută, dar servește la a prezenta modul de utilizare a clasei `FastTemplate` pentru a izola elementele de proiectare ale unui sit Web. Vă puteți baza pe acest exemplu simplu pentru a organiza structura unor situri Web mult mai mari și mai complexe.

### Test de evaluare

1. Specificați două avantaje ale utilizării șabloanelor pentru organizarea unui sit Web de mari dimensiuni.
2. Scrieți un bloc HTML care folosește o variabilă șablon numită `legatura` pentru a furniza adresa URL asociată unei legături. Textul asociat legăturii trebuie să fie „Duceți-vă acolo acum”.
3. Scrieți o instrucțiune PHP care asociază valoarea 3.14159 cu variabila șablon `pi` a clasei `FastTemplate`. Se va presupune că variabila PHP `$t` face referire la un obiect `FastTemplate`.
4. Scrieți o instrucțiune PHP care afișează valoarea asociată variabilei șablon `html` a clasei `FastTemplate`. Se va presupune că variabila PHP `$t` face referire la un obiect `FastTemplate`.
5. Scrieți o instrucțiune PHP care instanțiază un obiect `FastTemplate` ce folosește șabloanele stocate în catalogul părinte al catalogului care conține scriptul PHP. Stocați referința la obiect într-o variabilă PHP numită `$ft`.

## Modulul 17: Depanarea scripturilor PHP

### Scopuri

- Învățați să distingeți erorile și defectele
- Învățați să depanați erorile gramaticale
- Învățați să remediați erorile la rulare
- Învățați să stabiliți cu precizie defectele software

Mulți sunt de părere că dezvoltarea programelor de calculator reprezintă cel mai complex proces pe care l-a întreprins vreodată umanitatea. În consecință, nu trebuie să mire pe nimeni faptul că imperfecțiunile rasei umane joacă un anumit rol în dezvoltarea programelor. Programatorii trebuie să facă față unei diversități de surse de eroare, de manifestări și tipuri de erori, aplicând o gamă la fel de variată de instrumente și tehnici pentru a purta războiul cu erorile. Acest modul descrie erorile frecvent întâlnite de genul celor care apar în programele PHP, precum și tehnicile actuale pentru descoperirea, stabilirea cu precizie și eradicarea erorilor.

### Depanarea și erorile comune de programare

Depanarea este procesul de eliminare a hibelor de program, care reprezintă greșeli comise de programatori. Depanarea este diferită de testare, care constă în analiza unui program pentru a-i determina caracteristicile, cu precădere numărul și gravitatea defectelor pe care le conține. Practica modernă a testării programelor este o disciplină complexă, care depășește cadrul acestei cărți. Cititorul interesat în a afla mai multe despre testare este sfătuit să consulte cartea lui Boris Beizer *Black-Box Testing: Techniques for Functional Testing of Software and Systems*\* (Wiley, 1995) sau lucrarea lui Brian Marick *Craft of Software Testing: Subsystems Testing Including Object-Based and Object-Oriented Testing*\*\* (Prentice Hall, 1997).

Depanarea intră în scenă atunci când la testare apar simptome care indică existența unui defect. În general, depanarea implică

- Reproducerea simptomelor asociate hibeii
- Stabilirea cu precizie a locației hibeii
- Înțelegerea codului care conține hiba

\* În traducere *Testarea de tip „cutie neagră”: tehnici pentru testarea funcțională a programelor și a sistemelor* – N.T.

\*\* În traducere *Arta testării programelor: testarea subsistemelor, inclusiv testarea bazată pe obiecte și testarea orientată spre obiecte* – N.T.

- Remedierea hibe
- Testarea remedierii hibe și asigurarea că nu au apărut alte hibe, ca o consecință a remedierii

Secțiunea următoare, „Înțelegerea hibelor“, se concentrează asupra distingării a numeroase tipuri de erori frecvent întâlnite și sugerează metode de tratare a acestora. O secțiune ulterioară a acestui modul, în speță „Arta și practica depanării“, descrie în detaliu procesul de depanare.

## Notă

Din păcate, cuvântul *hibă* a ajuns să facă referire la o noțiune care poate fi descrisă într-un mod mai adecvat prin expresia *defect software*. De exemplu, același cuvânt poate face referire la un caz minor de gripă\*, care nu s-a produs din vina nimănui și nu pune problema unor consecințe. Similar, unii programatori denumesc defectele de program ca *hibe*, în încercarea de a le reduce la minimum și de a se eschiva de responsabilitatea comiterii sau remedierii acestora. Totuși, termenul a devenit atât de frecvent folosit, încât utilizarea altuia poate cauza confuzie.

## Înțelegerea hibelor

Când un programator comite o greșeală, atunci execută una sau mai multe din următoarele acțiuni:

- Omite liniile de program necesare
- Scrie linii de program inutile
- Scrie linii de program incorecte

Așa cum s-a arătat anterior, rezultatul este cunoscut sub numele de *hibă software* sau *hibă de program*.

Dovezile privind existența unei hibe pot fi obținute la rularea programului. Forma dominantă de testare a programelor, și anume testarea bazată pe execuție, implică rularea unui program cu intenția specifică de a găsi dovada existenței unei hibe. La rularea unui program PHP, dovada existenței unei hibe ia, de regulă, una din următoarele forme:

- Un mesaj PHP care indică o eroare de sintaxă
- Un mesaj PHP sau al bibliotecii PHP care indică o eroare la rulare
- Date de ieșire ale programului incorecte sau care lipsesc

Totuși, unele categorii de hibe nu prezintă asemenea dovezi. De exemplu, un program cu hibe care conține linii de program inutile poate rula mai lent sau mai puțin eficient decât un program corect. Totuși, programul eronat poate genera date

\* În jargonul calculatoarelor, *bug* înseamnă, într-adevăr, *hibă*. În limba engleză vorbită, *bug* mai înseamnă și *microb*, la această dublă semnificație se referă și autorul în cazul de față. – N.T.

de ieșire, respectiv rezultate corecte. Se spune că un asemenea program este *corect din punct de vedere funcțional*, deoarece execută funcția corectă, deși o execută cu performanțe slabe. Aproape toate operațiile de depanare sunt orientate spre remedierea programelor incorecte din punct de vedere funcțional, nu a programelor corecte sub acest aspect. Estimarea și îmbunătățirea performanței programelor corecte din punct de vedere funcțional constituie domeniul disciplinei cunoscută sub numele de *evaluare a performanței programelor*. Ca și testarea programelor, evaluarea performanței programelor este o disciplină specializată. Din păcate, disciplina de evaluare a performanței programelor este relativ puțin dezvoltată, deci nu putem recomanda cititorului interesat să consulte o literatură bine fundamentată referitoare la subiectul respectiv.

## Erori de sintaxă

Când interpretorul PHP încarcă un program PHP, îl analizează pentru a determina dacă programul se conformează regulilor gramaticale (sintactice) ale limbajului PHP. De exemplu, interpretorul verifică ortografia cuvintelor cheie și utilizarea oportună a caracterelor de delimitare, precum virgulele și caracterele punct și virgulă. Dacă programul nu se conformează gramaticii limbajului PHP, interpretorul afișează un mesaj de eroare sintactică. De exemplu, mesajul:

```
Parse error: parse error in test.php on line 3*
```

semnalează existența unei erori de sintaxă.

Cunoscătorii altor limbaje de programare decât PHP nu vor asocia imediat termenul de *hibă* cu noțiunea de eroare sintactică. Multe limbaje de programare impun programatorului să pregătească în mod special un program în vederea execuției, efectuând un proces cunoscut sub numele de *compilare*. Asemenea limbaje se numesc *limbaje compilate*; PHP și alte limbaje care nu necesită compilare se numesc *limbaje de scripting*.

Deși necesită un oarecare volum de efort suplimentar din partea programatorului, compilarea prezintă unele avantaje. În timpul compilării, erorile de sintaxă sunt descoperite și corectate. O dată un program pregătit pentru execuție, erorile de sintaxă nu mai sunt posibile. Astfel, la rularea unui program scris într-un limbaj compilat nu se poate genera o eroare de sintaxă.

Un program scris într-un limbaj de scripting poate genera erori de sintaxă la rulare. În consecință, în contextul respectiv, erorile de sintaxă pot fi considerate hibe, mai ales când cei care se confruntă cu acestea sunt utilizatorii programului, nu programatorii.

\* În traducere: *eroare de analiză: eroare de analiză în fișierul test.php în linia 3* – N.T.

Când interpretorul PHP raportează o eroare, acesta indică un număr de linie. Este important să înțelegem că numărul de linie semnalat nu precizează, în general, linia care conține eroarea; este vorba despre numărul liniei pe care interpretorul PHP o prelucra în momentul în care acesta a sesizat existența erorii. În consecință, eroarea de sintaxă se poate afla pe linia indicată sau anterior acesteia.

Uneori, eroarea de sintaxă se poate găsi cu multe linii înaintea liniei indicate. De exemplu, programatorii neglijează deseori să includă caracterul ghilimele care închide un șir de text. Într-un asemenea caz, interpretorul PHP poate considera că liniile care urmează după amplasamentul scontat al caracterului ghilimele inexistent fac de asemenea parte din șirul text. Când interpretorul PHP recunoaște, în sfârșit, că în program există o eroare, acesta poate indica spre o locație amplasată la zeci sau chiar sute de linii depărtare de poziția efectivă a erorii.

Pentru a evita erorile de sintaxă, este bine să vă corectați programul. Mai bine este să cereți unei alte persoane să vă corecteze programul. Pentru a stabili cu precizie poziția erorilor de sintaxă, folosiți tehnica *divide et impera*, prezentată în secțiunea intitulată „Arta și practica depanării”.

## Mesaje de eroare la rulare

O altă categorie de hibe PHP frecvent întâlnite este indicată de una sau mai multe mesaje de eroare la rulare. De exemplu, încercarea de împărțire la zero are ca rezultat un mesaj de eroare asemănător cu următorul:

```
Warning: Division by zero in test.php on line 2*
```

Acest mesaj este generat de interpretorul PHP, care este responsabil cu efectuarea calculelor. Similar, încercarea de a deschide un server de baze de date la o gazdă nepotrivită poate duce la un mesaj de eroare asemănător cu următorul:

```
Warning: MySQL Connection Failed: Unknown MySQL Server Host 'dbhost' (2) in test.php on line 3**.
```

Acest mesaj este generat de biblioteca MySQL a limbajului PHP, care este responsabilă cu interfața cu serverele de baze de date MySQL. Alte biblioteci PHP generează mesaje de eroare asociate operațiilor pe care le execută.

PHP generează patru nivele de mesaje de eroare. În ordinea crescătoare a gravității, acestea sunt următoarele:

- Anunțuri, care sunt trimise browserului numai dacă folosiți funcția `error_reporting()` pentru a specifica o sensibilitate la erori mai mare decât nivelul normal
- Erori de analizor, care indică o sintaxă incorectă a programului

\* În traducere Avertisment: împărțire la zero în fișierul `test.php`, în linia 2 – N.T.

\*\* În traducere Avertisment: Conexiune MySQL ratată: Gazdă necunoscută a serverului MySQL 'dbhost' în fișierul `test.php`, pe linia 3 – N.T.

- Avertismente, care provin deseori din erorile de domeniu
- Erorile fatale, care determină încheierea execuției scriptului

De regulă, mesajele de eroare sunt concepute pentru a veni în sprijinul programatorilor. Pentru utilizatori, acestea sunt deranjante sau chiar mai rău. De exemplu, un mesaj de eroare accidental poate afecta într-o asemenea măsură conținutul unui formular HTML generat de PHP, încât formularul devine inutilizabil. Deci, în condiții normale, un program corect din punct de vedere funcțional nu trebuie să genereze mesaje de eroare.

Dar nu toate condițiile sunt normale. Nu este lipsit de sens ca un program să genereze asemenea mesaje de eroare în condiții neobișnuite. În absența mesajelor de eroare, reproducerea unei probleme sau determinarea cauzelor acesteia sunt operații dificile sau imposibile. Astfel, o bună parte din proiectarea unei aplicații constă în a determina circumstanțele considerate „normale”, astfel încât mesajele de eroare să poată fi excluse sub aceste circumstanțe, dar permise în situații speciale. Sfaturile rigide – de genul eliminării permanente a mesajelor de eroare prin prefixarea numelor funcțiilor cu simbolul „coadă de maimuță” (@) – sunt simpliste și fără utilitate. Secțiunea următoare, intitulată „Gestiunea erorilor în PHP”, prezintă numeroase tehnici pentru tratarea mesajelor de eroare PHP.

Majoritatea mesajelor de eroare la rulare sunt rezultate ale erorilor de domeniu, în speță tentativelor de a aplica un operator sau o funcție unei valori necorespunzătoare. Împărțirea la zero este o eroare de domeniu frecvent întâlnită. Puteți evita frecvent mesajele de eroare la rulare prin simpla verificare a tipului și a valorii operanzilor și a argumentelor înainte de a le folosi.

Cu toate acestea, dușă la extrem, o asemenea strategie are ca rezultat programe de mari dimensiuni, ineficiente. O abordare mai rațională implică analiza riscurilor. Trebuie să verificați tipul și valoarea operanzilor și a argumentelor care pot fi frecvent incorecte sau care pot duce la probleme grave. Ca regulă empirică, este important să se verifice valorile introduse de utilizatori. În funcție de aplicație, este importantă și verificarea valorilor provenite din fișiere, baze de date sau surse externe.

PHP și alte limbaje de scripting prezintă probleme speciale, datorită tipurilor dinamice. Multe limbaje de programare impun specificarea tipului variabilei anterior utilizării acesteia, iar apoi asociază în mod permanent tipul respectiv cu variabila. Aceste limbaje se numesc *limbaje cu tipuri bine definite* (în original *strongly typed languages*). Deoarece tipurile variabilelor sunt cunoscute și fixate la compilare, unele categorii de erori de domeniu pot fi semnalate de compilatorul unui limbaj cu tipuri bine definite și implicit evitate la rulare. Limbaje precum PHP, dar și numeroase alte limbaje de scripting, se numesc *limbaje cu tipuri slab definite* (în original *weakly typed languages*). Definirea slabă a tipurilor permite începătorilor să scrie programe care

funcționează corect în majoritatea cazurilor. Cu toate acestea, definirea slabă a tipurilor întârzie până la momentul rulării recunoașterea unor erori de domeniu, ceea ce îngreunează scrierea de programe cu înaltă fiabilitate.

## Date de ieșire inexistente sau incorecte

Un alt tip frecvent întâlnit de hibă de program este semnalat atunci când un program produce date de ieșire incorecte sau nu produce datele de ieșire așteptate. Asemenea hibe se numesc *hibe logice*, deoarece rezultă, în general, dintr-o logică de program incorectă.

Logica de program implică trei elemente:

- **Secvență.** Este ordinea în care este executat programul
- **Selecție.** Reprezintă instrucțiunile care sunt executate și cele omise datorită instrucțiunilor condiționale, cum este instrucțiunea `if`
- **Iterație.** Numărul de execuții ale instrucțiunilor sub controlul buclelor, cum sunt buclele `for`

Evitarea în totalitate a erorilor din logica de program nu este omeneste posibilă. Corectarea programului poate duce la descoperirea multor asemenea erori și este un procedeu recomandabil. Disciplina ingineriei software, care este destinată a-i ajuta pe dezvoltatorii de programe să creeze produse de înaltă calitate, a definit o tehnică numită *inspecții software*, care este o formă extrem de eficientă de corectare a programelor. Pentru a învăța mai multe despre acest procedeu, consultați cartea lui Tom Gilb *Software Inspection* (Addison-Wesley, 1993).



### Sfatul specialistului

**Întrebare:** Am folosit limbaje de programare, precum Microsoft Visual Basic, care includ un program de depanare. PHP conține un asemenea program?

**Răspuns:** PHP include un program de depanare în rețea. Totuși, programul de depanare nu a fost încă portat la PHP 4. Pentru a depana un program PHP, puteți folosi procedeele descrise în secțiunea intitulată „Arta și practica depanării”.



### Test „la minut”

- Când PHP semnalează o eroare de sintaxă, unde se află eroarea efectivă în raport cu numărul de linie indicat?
- Cum se numește o eroare relativă la valoarea sau tipul unui operand sau al unui argument?

- Care este metoda de definire a tipurilor frecvent folosită de limbajele de scripting, precum PHP?

## Gestiunea mesajelor de eroare în PHP

Unele limbaje folosite pentru dezvoltarea în Web, cum ar fi ColdFusion, Java și Python – furnizează mecanisme de tratare a excepțiilor. Mecanismele de tratare a excepțiilor vă permit să scrieți programe care primesc automat controlul la apariția unei erori. Programele dumneavoastră de tratare a erorilor pot, de exemplu, să raporteze eroarea și să încheie execuția programului sau pot încerca să ocolească eroarea și să-și continue execuția.

În prezent, PHP nu dispune de mecanisme pentru tratarea excepțiilor, deși aceasta este o caracteristică pe care mulți programatori se așteaptă să o vadă adăugată, într-un târziu, în limbajul respectiv. Până atunci, aveți la dispoziție trei tehnici de bază pentru tratarea mesajelor de eroare PHP:

- Evitarea condițiilor de eroare care ar genera, în caz contrar, mesaje de eroare
- Suprimarea mesajelor de eroare
- Consemnarea în jurnal a mesajelor de eroare

Cele trei procedee vor fi descrise în sub-secțiunile următoare.

## Evitarea mesajelor de eroare

Așa cum s-a explicat în secțiunea anterioară, multe mesaje de eroare PHP reprezintă rezultatul unor erori de domeniu. Puteți evita asemenea mesaje de eroare prin scrierea de programe care verifică valoarea și tipul operanzilor și al argumentelor înainte de utilizarea acestora. Totuși, când detectează o valoare sau un tip inadecvat, programul dumneavoastră trebuie să rezolve situația într-un fel sau altul.

Un procedeu comun constă în stabilirea unei funcții speciale pentru manipularea erorilor. Când programul detectează o eroare, invocă funcția de tratare a erorilor. La rândul său, funcția de tratare a erorilor poate executa oricare din următoarele operații:

- Poate afișa un mesaj de eroare prietenos cu utilizatorul
- Poate consemna eroarea într-un fișier sau într-o bază de date
- Poate încerca să ocolească eroarea

Răspunsuri la test:

- Pe linia respectivă sau anterior acesteia
- Eroare de domeniu
- Definire slabă



Centralizarea metodelor de tratare a erorilor într-o asemenea funcție prezintă numeroase avantaje. De exemplu:

- Facilitează implementarea a numeroase moduri de raportare a erorilor
- Simplifică adaptarea mesajelor de eroare pentru mai multe limbi sau localizări

Existența mai multor moduri de raportare a erorilor vă permite să configurați un program care să producă mesaje de eroare în faza de dezvoltare a programului, dar care să le suprimă în timpul operării programului. Astfel, programatorii pot beneficia de informațiile incluse în mesajele de eroare, dar utilizatorii pot evita aceste mesaje, care îi pot deruta sau stânjeni.

Similar, adaptarea unui program de așa natură încât să furnizeze mesaje orientate spre utilizator în mai multe limbi este mai simplă dacă o singură funcție, respectiv grup de funcții corelate, tratează toate condițiile de eroare. Mesajele orientate spre utilizator pot fi afișate într-o locație bine determinată pe ecran, astfel încât să nu afecteze funcționarea unei aplicații. De asemenea, aceste mesaje pot include informații codificate, care îi ajută pe programatori să identifice și să depaneze erorile. Astfel, aceste mesaje pot furniza informații echivalente celor conținute în mesajele de eroare PHP, fără a deruta sau devia atenția utilizatorului.

## Suprimarea mesajelor de eroare

Deseori, suprimarea mesajelor de eroare constituie cea mai simplă modalitate care se poate aplica. Dar, așa cum s-a explicat, această metodă refuză programatorilor accesul la informații care ar putea ajuta la „deconspirarea” existenței unei hibe sau la depanarea unei hibe cunoscute.

Mai mult, erorile PHP fatale determină încheierea execuției scriptului, chiar dacă mesajele de eroare sunt suprimate. Deci, în general este necesar să se testeze existența condițiilor de apariție a erorilor, chiar și atunci când mesajele de eroare sunt suprimate. Deseori, este doar puțin mai dificil să se trateze condiția de apariție a erorii folosind o funcție de tratare a erorilor, așa cum s-a arătat în secțiunea anterioară, sau prin consemnarea în jurnal a mesajelor de eroare, așa cum se va vedea în secțiunea următoare.

Puteți suprima mesajele de eroare într-unul din două moduri. Pentru a suprima mesajele de eroare în general, invocați funcția `error_reporting()`, care preia un argument ce specifică nivelul dorit de raportare a erorilor. Fiecare nivel de eroare are o valoare integrală asociată:

Nivel	Valoare asociată
Erori fatale	<code>E_ERROR</code>
Avertismente	<code>E_WARNING</code>
Erori de analizor	<code>E_PARSE</code>
Anunțuri	<code>E_NOTICE</code>

Argumentul transferat funcției `error_reporting()` este suma valorilor asociate nivelurilor care urmează a fi raportate. De exemplu, o valoare a argumentului egală cu `E_ERROR+E_WARNING` va determina PHP să raporteze numai erorile fatale și avertismentele. O valoare a argumentului egală cu zero va determina PHP să nu raporteze nici o eroare.

Alternativ, puteți suprima mesajele de eroare asociate apelului la o anumită funcție. Pentru aceasta, prefixați numele funcției cu simbolul „coadă de maimuță” (`@`). De exemplu, următorul apel de funcție nu va produce mesaje de eroare:

```
$db = @mysql_connect("localhost");
```

Atunci când mesajele de eroare sunt suprimate folosind oricare dintre tehnicile prezentate, puteți dori să obțineți acces la mesajul de eroare care ar fi fost afișat în caz de neutilizare a procedurilor. Textul mesajului este inclus în variabila specială `php_errormsg()`. Puteți folosi valoarea acestei variabile pentru a genera sau pentru a consemna în jurnal propriile dumneavoastră mesaje de eroare.

## Consemnarea mesajelor de eroare

Administratorul de sistem PHP poate configura PHP astfel încât să accepte consemnarea automată a mesajelor de eroare într-un fișier jurnal desemnat. Acesta este un obicei recomandat, deoarece evită imixtiunea mesajelor de eroare în datele de ieșire destinate utilizatorilor, dar în același timp capturează mesajele de eroare, pentru a fi analizate de programatori.

Pentru a configura PHP astfel încât să accepte consemnarea automată a mesajelor de eroare, administratorul de sistem trebuie să activeze următoarele opțiuni de configurare PHP în fișierul `php.ini`:

```
log_errors = On
error_log = fisier
```

unde `fișier` precizează calea spre fișierul jurnal care urmează a fi utilizat. În locul unui fișier, poate fi specificată în schimb valoarea specială `syslog`; această valoare determină consemnarea mesajelor PHP folosind jurnalul de sistem standard. Modificările aduse acestei opțiuni intră în vigoare la următoarea pornire a serverului Web.

În afară de consemnarea automată a mesajelor de eroare, PHP poate consemna manual mesajele de eroare specificate de funcția `error_log()`. PHP 4 acceptă trei forme ale acestei funcții:

- Mesajele de eroare sunt consemnate conform celor specificate de articolul de configurare `error_log`
- Mesajele de eroare sunt consemnate prin expedierea unor mesaje de e-mail
- Mesajele de eroare sunt consemnate prin scrierea într-un fișier specificat

Pentru a scrie un mesaj de eroare în jurnalul sistem PHP, invocați funcția:

```
error_log(mesaj, 0);
```

unde *mesaj* specifică mesajul care va fi consemnat. De exemplu, instrucțiunea:

```
error_log("Toate bune cu PHP.", 0);
```

scrie mesajul „Toate bune cu PHP.” la destinația specificată de articolul de configurare PHP `error_log`.

Pentru a expedia un mesaj de eroare prin intermediul poștei electronice, invocați funcția:

```
error_log(mesaj, 1, destinatie);
```

unde *mesaj* specifică mesajul care va fi consemnat, iar *destinatie* indică adresa de destinație a mesajului de e-mail. De exemplu, instrucțiunea următoare trimite un mesaj de eroare prin intermediul poștei electronice:

```
error_log("PHP functioneaza.", 1, "bill@osborne.com");
```

Dacă trebuie să specificați alte antete de mesaj, puteți folosi o formă conexă a funcției:

```
error_log(mesaj, 1, destinatie, extra);
```

unde *mesaj* specifică mesajul care va fi consemnat, *destinatie* indică adresa de destinație a mesajului de e-mail, iar *extra* specifică antetele de mesaj suplimentare.

Pentru a scrie un mesaj de eroare într-un fișier jurnal, invocați funcția:

```
error_log(mesaj, 3, destinatie);
```

unde *mesaj* specifică mesajul care va fi consemnat, iar *destinatie* indică adresa de destinație a mesajului de e-mail. De exemplu, instrucțiunea următoare trimite un mesaj de eroare într-un fișier jurnal specific aplicației:

```
error_log("A-Okay.", 3, "/var/log/aplmea.log");
```

Așa cum se poate observa, contul sub care rulează PHP trebuie să aibă acces de scriere în fișierul jurnal specificat.

## ? Sfatul specialistului

**Întrebare:** Și dacă doresc să consemnez date, în locul mesajelor de eroare? Are PHP această posibilitate?

**Răspuns:** Puteți folosi opțiunea funcției `error_log()` care permite consemnarea într-un anumit fișier pentru a scrie date arbitrare, nu doar mesaje de eroare. Pur și simplu reprezentați datele pe care doriți să le consemnați sub forma unui șir și transferați șirul funcției `error_log()`, alături de codul de destinație (3) și de calea spre fișierul jurnal. Puteți folosi această tehnică pentru a consemna accesele la părți ale sitului dumneavoastră Web, pentru a crea jurnale de tranzacție, pentru a consemna conexiunile și deconectările utilizatorilor de la rețea și pentru aproape absolut orice altceva.



## Test „la minut”

- Cum se numește nivelul de mesaje de eroare PHP de maximă gravitate?
- Dacă suprimați mesajele de eroare, este totuși posibil ca o eroare să determine încheierea execuției scriptului?
- Care este funcția PHP folosită pentru scrierea mesajelor de eroare adaptate?

## Arta și practica depanării

Această secțiune prezintă unele dintre principiile și tehnicile aplicabile în depănarea programelor. Așa cum o arată și titlul secțiunii, depănarea este mai mult o artă decât o știință. În consecință, depănarea este un proces bazat pe oportunitate: nu contează care este cea mai bună metodă de depănare a unui program, ci numai ca metoda respectivă să dea rezultatul scontat. Existența unor abordări și a unor perspective suplimentare poate contribui la îmbunătățirea eficienței procesului de depănare.

Secțiunea de față este organizată în jurul procesului de depănare specificat anterior, care include următoarele operații:

- Reproducerea simptomului
- Stabilirea cu precizie a hibe
- Înțelegerea hibe
- Remedierea hibe
- Testarea programului

Rețineți că operațiile sunt distincte de etapele prezentate. Operațiile pot fi executate într-o altă ordine, omise sau se pot chiar suprapune. Totuși, în general, operațiile specificate se execută succesiv, iar expunerea sugerează unele motive pentru care omiterea unei operații sau executarea acesteia în afara secvenței pot afecta eficacitatea procesului de depănare.

## Reproducerea simptomului

Prima activitate din cadrul procesului de depănare este reproducerea, în circumstanțe controlate, a simptomului sau simptomelor care indică existența hibe. Deseori, presupusele erori se dovedesc a fi o lipsă de înțelegere a modului de comportare a unui program sau mai degrabă o eroare a utilizatorului, nu a progra-

Răspunsuri la test:

- Eroare fatală
- Da
- `error_log()`

matorului. Deci, este important să încercați a reproduce simptomele pentru a verifica probabilitatea de existență a unei hibe.

Un al doilea motiv pentru reproducerea simptomului este acela că, în caz contrar, nu se poate demonstra că hiba a fost remediată. Dacă un simptom apare rar și numai în condiții care nu sunt foarte bine înțelese, s-ar putea să nu fie clar dacă hiba a dispărut sau doar se ascunde.

Mulți programatori dau fuga la următoarea operație, în speță stabilirea cu precizie a hibe. Totuși, procesul de reproducere a simptomului poate furniza indicii utile pentru stabilirea exactă a naturii hibe. De exemplu, este important să cunoaștem condițiile în care apare simptomul, precum și circumstanțele în care acesta nu apare.

La reproducerea simptomului, este util să dispunem de un raport complet și exact al circumstanțelor în care a fost observat simptomul. În caz contrar, încercarea de a reproduce simptomul poate eșua.

## Stabilirea cu precizie a hibe

O dată devenită posibilă reproducerea în condiții de siguranță a simptomului care indică existența unei hibe, se poate începe operația de stabilire cu precizie a hibe. În acest sens, două procedee sunt deosebit de utile: *divide et impera*\* și urmărirea programului.

## Descoperirea defectelor cu ajutorul tehnicii *divide et impera*

Tehnica *divide et impera* se bazează pe posibilitatea ca hiba să fie localizată într-o anumită secțiune a programului, ceea ce constituie o caracteristică specifică multor hibe, poate majorității hibelor care se găsesc în programele PHP. Procedeu *divide et impera* implică eliminarea liniilor de program selectate și rularea programului. Dacă simptomul persistă, s-a demonstrat că hiba se află în cealaltă porțiune a programului.

Procedeu *divide et impera* poate fi extrem de eficient. Dacă o jumătate a programului este eliminată la fiecare iterație a acestei metode, o hibă care afectează o singură linie dintr-un program cu 1000 de linii poate fi localizată prin numai zece iterații. Totuși, acest procedeu este rareori folosit până la identificarea unei singure linii de program. În schimb, este frecvent utilizat pentru a detecta o funcție cu defect. Dacă funcțiile unui program au o lungime medie de 25 de linii, un program cu 1000 de linii constă din 40 de asemenea funcții. Astfel, pentru identificarea funcției afectate vor fi necesare numai cinci sau șase iterații ale metodei *divide et impera*.

În mod caracteristic, liniile eliminate în cursul aplicării procedurii nu sunt eliminate din fișierul program; în schimb, sunt transformate în comentarii. Cu alte cuvinte, sunt plasate marcare de comentariu astfel încât interpretorul PHP să considere liniile eliminate ca fiind comentarii, nu linii de program executabile.

\* Dicton latin: *dezbina și stăpânește* – N.T.

O complicație la aplicarea procedurii *divide et impera* constă în faptul că există corelații între componentele a numeroase programe, astfel că eliminarea unui program va cauza eșuarea unei părți corespondente a programului. În acest caz, se poate folosi o variantă specială a procedurii *divide et impera*, cunoscută sub numele de *racorduri și drivere* (în original *stubs and drivers*). Procedeu *racorduri și drivere* folosește racordurile pentru a aplica tehnica *divide et impera* unui program, iar driverele pentru a executa celelalte componente ale programului.

*Racord* este termenul folosit pentru a desemna o funcție intenționat incompletă. A *racorda* o funcție înseamnă a înlocui funcția cu un program care afișează un mesaj sau care returnează o valoare fixă, fără a executa o operație efectivă. De exemplu, iată un racord simplu ce poate înlocui o funcție care calculează rădăcina cubică a argumentului său:

```
function radacina_cubica(x)
{
 echo "
radacina_cubica() apelata cu argumentul $x";
 return 1;
}
```

Remarcați că funcția *racord* returnează întotdeauna valoarea 1. În general, nu acesta este rezultatul matematic corect; totuși, este puțin probabil ca valoarea 1 să determine eșecul funcțiilor care invocă funcția *radacina\_cubica()*.

Prin *driver* se înțelege o funcție care apelează o altă funcție, transferând argumente speciale destinate a scoate la iveală hibe ascunse în cadrul acesteia din urmă. De exemplu, iată o funcție *driver* care se poate folosi pentru testarea unei funcții numite *patru()* care calculează rădăcini de ordinul al patrulea:

```
function driver($x)
{
 echo "
Apeleaza functia patru() cu argumentul $x:";
 $y = patru($x);
 echo "
Funcția patru() a returnat rezultatul $y.";
 $z = $y * $y * $y * $y;
 if ($x == $z)
 echo "
Rezultatul este corect.";
 else
 echo "
Rezultatul nu este corect.";
}
```

## Urmărirea evoluției unei variabile

Deseori, verificarea modului de funcționare a unui program este mai simplă dacă sunt disponibile rezultate intermediare. De exemplu, un program care determină dacă un număr dat este prim poate fi depanat mai ușor dacă se poate vedea care au fost factorii testați.

Unele limbaje de programare includ utilitare de depanare care permit specificarea unei opțiuni ce determină afișarea linie cu linie a execuției programului. PHP nu furnizează o asemenea caracteristică. Totuși, puteți folosi instrucțiuni *echo* pentru a

afișa mesaje care prezintă evoluția unui proces de prelucrare a datelor și valorile variabilelor importante.

Practica inserției unor asemenea instrucțiuni într-un program se numește *instrumentarea* programului. Unii programatori își instrumentează programele în faza de dezvoltare și nu elimină elementele de instrumentare nici măcar când programele intră în faza de producție. De obicei, se folosesc instrucțiuni `if` care suprimă datele de ieșire instrumentate dacă o variabilă globală – denumită deseori `$debug` (depanare) sau ceva asemănător – nu are o anumită valoare.

Acest obicei este deosebit de util în lucrul cu programe care s-au dovedit a fi generatoare de probleme. Când este necesară depanarea programului, nu trebuie să-l instrumentați, deoarece procedeul a fost deja aplicat.

## Înțelegerea hibe

Operația următoare, adică înțelegerea hibe, este cel mai frecvent neglijată de către programatori. Deseori, programatorii încep să efectueze revizuirii prin încercări, în speranța că vor descoperi o corecție care va remedia problema. O asemenea abordare aleatoare, de tip „învățare din greșeli”, a procesului de depanare se dovedește rareori oportună, deoarece programatorii care procedează astfel deseori introduc hibe noi în timp ce încearcă să le remedieze pe cele vechi.

Este esențial să înțelegeți natura hibe înainte de a încerca să o remediați. Dacă programul este prea complicat pentru a fi înțeles, acesta este un indiciu că procedeul *divide et impera* a fost abandonat prematur. Dacă este necesar, izolați instrucțiunea individuală „responsabilă” cu hiba înainte de a încerca să o remediați. Majoritatea programatorilor – chiar și începătorii – sunt capabili de a înțelege complet o instrucțiune separată de restul programului.

## Remedierea hibe

În general, remedierea hibe este cea mai simplă operație din cadrul procesului de depanare. Totuși, poate constitui și o oportunitate ratată. Deseori, o hibă poate fi remediată în mai multe moduri. Nu alegeți pur și simplu prima soluție la care v-ați gândit, ci încercați să determinați și care sunt celelalte remedii posibile. Apoi, dintre acestea, alegeți soluția cea mai bună, nu neapărat pe prima.

## Testarea programului

Operația finală din cadrul procesului de depanare este cea mai importantă. Mulți programatori – mai ales începătorii – își revizuiesc programele fără a verifica dacă varianta revizuită rezolvă efectiv problema. Verificarea dispariției simptomului este o componentă necesară a procesului de depanare. Totuși, această operație nu este

suficientă. Deseori, în procesul de depanare, sunt introduse hibe noi. Este important să verificați două aspecte ale programului revizuit:

- Dacă operațiile anterior imposibile sunt acum posibile
- Dacă operațiile posibile anterior sunt și acum posibile

Satisfacerea ambelor criterii este dificilă. O metodă constă în dezvoltarea unor cazuri de test prin regresie pentru fiecare program. În acest context, un *test prin regresie* este un test sau o serie de teste rulate la fiecare modificare a unui program. Inițial, este posibil ca setul de cazuri de teste prin regresie să nu fie foarte cuprinzător. Dar, dacă se acumulează cazuri de test care tratează fiecare eroare care apare, setul de cazuri de teste prin regresie se va dovedi, în cele din urmă, util pentru evitarea hibelor care, în cazul neutilizării acestui procedeu, ar fi fost introduse în programul difuzat pentru public. Disciplina de testare a programelor furnizează principii utile pentru alegerea cazurilor de teste prin regresie; cititorul interesat va consulta referințele specificate anterior.



### Sfatul specialistului

**Întrebare:** N-am timp să devin expert în testare. Nu aveți câteva sfaturi rapide privind modul de alegere a cazurilor de test?

**Răspuns:** O tehnică de testare utilă se numește *partiționare prin echivalență*.

Acest procedeu pornește de la observația că multe cazuri de test sunt cruciale în sensul că, dacă funcționează, atunci vor funcționa garantat și numeroase alte cazuri de test conexe. Prin testarea unui număr maxim permis în practică din aceste cazuri de test cruciale, testarea programelor devine mai eficientă decât atunci când cazurile de test sunt selectate în mod arbitrar.

O euristică utilă pentru partiționarea prin echivalență constă în testarea unor valori mici, medii și mari pentru fiecare câmp prelucrat de un program. Pentru numere, testați valori pozitive, negative, respectiv egale cu zero. Pentru câmpurile care apar în formularele HTML, testați valorile critice, precum valorile asociate casetelor de validare, butoanelor radio și controalelor de selectare.

Dacă testați pur și simplu aceste valori pentru fiecare câmp de introducere a datelor, testarea programului efectuată de dumneavoastră va fi, probabil, mai eficientă decât aceea executată de un programator mediu.



### Test „la minut”

- Care este prima operație care trebuie executată în cadrul procesului de depanare?
- Care este operația din cadrul procesului de depanare cel mai frecvent omisă de programatorii începători?

- Care este cea mai simplă operație din cadrul procesului de depanare?



### Proiect 17-1: Mesaje de eroare PHP

În cadrul acestui proiect, veți vedea „la prima mână” cum se comportă mesajele de eroare PHP. Veți folosi funcția `error_reporting()` pentru a specifica nivelul de raportare a erorilor și pentru a genera erorile pe care PHP le raportează sau le ignoră, în funcție de nivelul curent de raportare a erorilor.

#### Scopurile proiectului

- Prezentarea modului de utilizare a funcției `error_reporting()`
- Prezentarea influenței exercitate de nivelul de raportare a erorilor asupra modalității de raportare a mesajelor de eroare

#### Pas cu pas

1. Creați următorul script HTML, plasați-l într-un fișier numit `p17-1.php` și încărcați-l în serverul dumneavoastră PHP:

```
<HTML>
<HEAD>
<TITLE>Proiect 17-1</TITLE>
</HEAD>
<BODY>
<?php

echo "
Utilizeaza raportarea standard a erorilor:";

$x = $y;
$x = $ x / 0;

echo "
Activeaza raportarea tuturor erorilor:";
error_reporting(15);

$x = $y;
$x = $ x / 0;

echo "
Dezactiveaza raportarea erorilor:";
error_reporting(0);

$x = $y;
$x = $ x / 0;

echo "

Genereaza o eroare fatala:";
$x = aceasta_functie_nu_este_definita();

echo "
Ultima linie.";
```

#### Răspunsuri la test:

- reproducerea simptomului
- înțelegerea hibe
- remedierea hibe

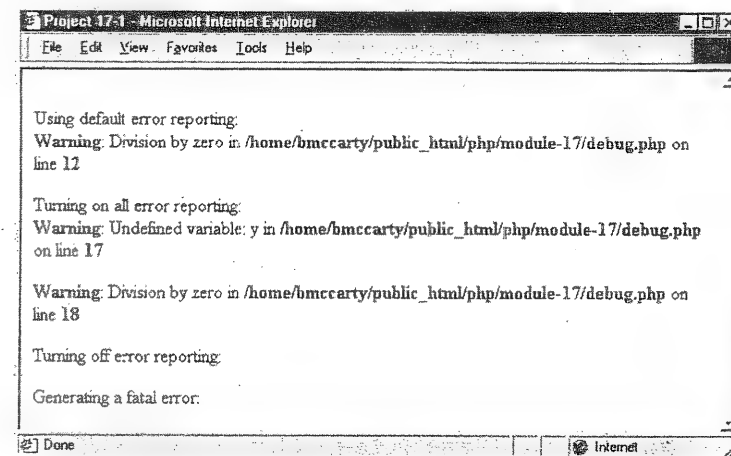
```
?>
</BODY>
</HTML>
```

2. Studiați scriptul pentru a discerne elementele de structură a programului. Veți identifica următoarele elemente:

- O secțiune de program care folosește nivelul prestabilit de raportare a erorilor
- O secțiune de program care folosește nivelul cel mai detaliat de raportare a erorilor
- O secțiune de program care suprimă toate mesajele de eroare
- O secțiune de program care apelează o funcție nedefinită și care execută apoi o instrucțiune `echo`

Fiecare din primele trei elemente ale structurii include un program care generează două erori: o referință la o variabilă nedefinită și o împărțire la-zero. Prima este considerată o eroare la nivel de anunț, iar ultima este o eroare de nivel avertisment. Ultimul element al structurii apelează o funcție nedefinită; aceasta este considerată ca fiind o eroare fatală.

3. Orientați browserul dumneavoastră Web spre adresa URL asociată scriptului PHP `p17-1.php`. Ecranul browserului va fi asemănător cu următoarea ilustrație:



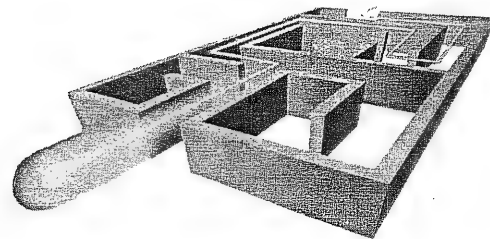
4. Observați că:

- La nivelul prestabilit de raportare a erorilor, nu sunt afișate anunțurile, dar avertismentele sunt afișate
- La nivelul cel mai detaliat de raportare a erorilor, sunt afișate atât anunțurile, cât și avertismentele
- Când raportarea erorilor este suprimată, nu sunt afișate nici anunțurile, nici avertismentele
- Când raportarea erorilor este suprimată, nu sunt afișate nici măcar erorile fatale, deși acestea au ca rezultat încheierea imediată a execuției programului, așa cum s-a arătat prin lipsa datelor de ieșire ale instrucțiunii `echo` finale



### Test de evaluare

1. Care este numele tehnicii de depanare care implică și transformarea în comentarii a unor secțiuni de program?
2. Include PHP 4 o funcționalitate de depanare?
3. Care este variabila PHP ce include textul celui mai recent mesaj de eroare, chiar dacă raportarea erorilor este suprimată?
4. Cum se numesc erorile care încalcă regulile gramaticale ale limbajului PHP?
5. Ce se întâmplă la apariția unei erori fatale, atunci când raportarea erorilor este suprimată?



## Partea a V-a Anexe

### Anexa A: Răspunsuri la testele de verificare

#### Modulul 1: Crearea programelor PHP

1. Ce program Windows este frecvent folosit pentru crearea scripturilor PHP?

Notepad

2. Care trebuie să fie prima linie într-un script PHP?

<?php

3. Care sunt caracterele ce trebuie folosite pentru a denumi un fișier care conține un script PHP?

Litere scrise cu minuscule, cifre și caractere de subliniere

4. Care trebuie să fie extensia unui fișier care conține un script PHP?

.php

5. Care este instrucțiunea PHP folosită pentru a trimite date de ieșire sub formă de text unui browser Web?

echo

6. Care este programul frecvent folosit pentru a încărca un script PHP într-un server?

FTP



## Modulul 2: Elementele constructive ale limbajului PHP

1. Scrieți o valoare PHP literală egală cu 12000.  
12000
2. Scrieți o valoare PHP literală egală cu  $10^{39}$ .  
1.0e39
3. Scrieți o valoare PHP literală care conține numele mărcii autoturismului preferat.  
"mercedes"  
(sau similar)
4. Scrieți numele unei variabile PHP adecvate pentru stocarea ratei impozitului aferent vânzărilor curente.  
\$rata\_impozit\_vanzari  
(sau similar)
5. Scrieți instrucțiuni PHP care creează un tablou ce asociază numele de botez al fiecăruia dintre membrii familiei dumneavoastră cu anul în care s-a născut persoana respectivă.  
\$an["radu"] = 1933;  
\$an["mariana"] = 1940;  
(și așa mai departe, sau similar)
6. Scrieți o instrucțiune PHP care calculează circumferința unui cerc pornind de la raza sa, dacă este cunoscută ecuația matematică  $C = 2\pi R$  și valoarea aproximativă a lui  $\pi$  egală cu 3,14159.  
\$circumferinta = 3.14159 \* 2 \* \$raza;  
(sau similar)
7. Scrieți o instrucțiune PHP care calculează valoarea absolută a variabilei \$distanța și stochează rezultatul în variabila \$distanța\_neta.  
\$distanța\_neta = abs(\$distanța);  
(sau similar)

## Modulul 3: Crearea formulelor HTML

1. Scrieți o etichetă HTML FORM care își trimite datele unui script situat la adresa <http://www.osborne.com/cgi-bin/test>.  
<FORM METHOD="POST"  
ACTION="http://www.osborne.com/cgi-bin/test">
2. Scrieți un program HTML care creează un control cu mai multe linii, denumit adresa, pentru introducerea de text. Controlul trebuie să poată conține 5 rânduri a 80 de caractere fiecare.  
<TEXTAREA NAME="adresa" ROWS="5" COLS="80" WRAP="xxx">

3. Scrieți un program HTML care creează un meniu derulant denumit culoare, care conține principalele culori substructive, în speță roșu, galben și albastru. Faceți de așa manieră încât meniul să accepte o singură selecție. Specificați culoarea roșie ca opțiune prestabilită.  
<SELECT NAME="culoare" SIZE="1">  
<OPTION SELECTED>Rosu  
<OPTION>Galben  
<OPTION>Albastru  
</SELECT>
4. Scrieți un program HTML pentru crearea unui set de butoane radio denumite dimensiune, care permit utilizatorului să aleagă din următoarele valori: mic, mediu și mare. Butoanul aferent valorii „mare” va fi selectat în mod prestabilit.  
<INPUT TYPE="RADIO" NAME="dimensiune" VALUE="Mic">  
<INPUT TYPE="RADIO" NAME="dimensiune" VALUE="Mediu">  
<INPUT TYPE="RADIO" NAME="dimensiune" VALUE="Mare" CHECKED>
5. Scrieți un program HTML pentru crearea unui formular care își trimite datele la adresa [www.dev.null](http://www.dev.null). Formularul trebuie să conțină un câmp ascuns denumit script, care conține meniul cu valori.  
<FORM METHOD="POST" ACTION="www.dev.null">  
<INPUT TYPE="HIDDEN" NAME="script" VALUE="menu">  
<INPUT TYPE="SUBMIT">  
</FORM>

## Modulul 4: Accesul la date

1. Care este variabila PHP ce trebuie folosită pentru a obține accesul la datele asociate unui control definit de eticheta HTML <INPUT TYPE="TEXT" NAME="culoare">?  
\$culoare
2. Scrieți o instrucțiune PHP care trimite browserului valoarea variabilei \$x.  
echo "\$x";
3. Scrieți o instrucțiune PHP care trimite browserului numele variabilei \$y.  
echo "\\$y";
4. Scrieți o instrucțiune PHP care trimite browserului adresa URL a paginii care face referire la pagina curentă.  
echo "\$HTTP\_REFERER";

## Modulul 5: Lucrul cu valori scalare

1. Scrieți o instrucțiune care definește o constantă denumită VITEZA, care are valoarea 186.282\*.  

```
define("VITEZA", 186282);
```
2. Scrieți o instrucțiune care afișează o valoare ce indică dacă a fost sau nu definită constanta LUNGIME.  

```
echo defined("LUNGIME");
```
3. Dacă variabila \$pisica are valoarea "Tom" și dacă variabila \$animal are valoarea "pisica", care este numele unei variabile dinamice cu valoarea "Tom"?  

```
$$animal
```
4. Dacă se procedează la împărțirea a două valori întregi, care este tipul rezultatului?  

Întreg
5. Dacă o valoare de tip întreg se împarte la o valoare de tip dublu, care este tipul rezultatului?  

Dublu
6. Scrieți o instrucțiune care modifică tipul variabilei \$cost în întreg.  

```
settype($cost, "integer");
sau $cost = (integer) $cost;
```

## Modulul 6: Scrierea instrucțiunilor condiționale

1. Scrieți o instrucțiune if care atribuie variabilei \$y valoarea 1 dacă variabila \$x are valoarea 1, în caz contrar atribuind variabilei \$y valoarea 2.  

```
if ($x == 1)
 $y = 1;
else
 $y = 2;
```
2. Scrieți o instrucțiune switch care atribuie valoarea 5 variabilei \$y dacă variabila \$x are valoarea 1, respectiv valoarea 15 dacă variabila \$x are valoarea 2, valoarea 20 dacă variabila \$x are valoarea 3, valoarea -1 în celelalte situații.  

```
switch($x)
{
 case 1:
 $y = 5;
 break;
 case 2:
 $y = 15;
 break;
```

\* Este vorba despre viteza luminii, exprimată în mile pe secundă. – N. T.

- ```
case 3:  
    $y = 20;  
    break;  
default:  
    $y = -1;
```
3. Scrieți o buclă for care are ca date de ieșire o serie de asteriscuri; numărul asteriscurilor trebuie să fie dat de valoarea variabilei \$stele.

```
for ($i = 1; $i <= $stele; $i++)  
    echo "**";
```
 4. Scrieți o instrucțiune if care atribuie variabilei \$y valoarea 1 dacă variabila \$x are valoarea 1, respectiv valoarea 3 dacă variabila \$x are valoarea 2, valoarea 5 dacă variabila \$x are valoarea 3, valoarea -1 în celelalte situații.

```
if ($x == 1)  
    $y = 1;  
elseif ($x == 2)  
    $y = 3;  
elseif ($x == 3)  
    $y = 5;  
else  
    $y = -1;
```

Modulul 7: Utilizarea funcțiilor

1. Scrieți o instrucțiune care invocă funcția test(), transferând valorile 1 și 2 ca argumente.

```
test(1,2);
```
2. Scrieți o instrucțiune care invocă funcția live(), transferând ca argumente valorile 1 și 2; asigurați-vă că nu se vor genera mesaje de eroare în timpul execuției funcției.

```
@live(1,2);
```
3. Scrieți o instrucțiune care include conținutul fișierului antet.php ca parte a scriptului curent.

```
require("antet.php");
```
4. Scrieți definiția unei funcții numite patrat(), care calculează aria unui pătrat, dacă este dată lungimea unei laturi a pătratului.

```
function patrat($latura)  
{  
    return $latura * $latura;  
}
```
5. Scrieți o definiție a unei funcții denumite contor(), care incrementează și returnează valoarea unei variabile locale statice.

```
function contor()  
{
```

```
static contor;
contor++;
return contor;
}
```

Modulul 8: Utilizarea tablourilor

1. Scrieți instrucțiuni care creează un tablou denumit \$pop, care asociază numele mai multor orașe mari cu numărul locuitorilor acestora.

```
$pop["Tokio"] = 34500000;
$pop["New York"] = 20200000;
etc.
```

2. Scrieți o instrucțiune for care parcurge în mod iterativ un tablou secvențial denumit \$pitici, unde cheia minimă are valoarea unu. Corpul instrucțiunii for trebuie să afișeze numele fiecărui element al tabloului \$pitici. Aveți grijă la scrierea expresiei de test, care trebuie să reflecte faptul că valoarea cea mai mică a unei chei este unu, nu zero.

```
$n = count($pitici);
for ($i = 1; $i <= $n; $i++)
{
    echo "<BR>$pitici[$i]";
}
```

3. Scrieți o instrucțiune foreach care caută în tabloul \$state un element a cărui cheie are aceeași valoare ca și variabila \$abrev. Afișați valoarea elementului corespunzător, nu cheia acestuia.

```
foreach ($state as $cheie => $valoare)
{
    if ($valoare == $abrev)
    {
        echo "<BR>$valoare";
        break;
    }
}
```

4. Scrieți o instrucțiune care sortează tabloul asociativ \$pop în ordine crescătoare, în funcție de valoare.

```
asort($pop);
```

Modulul 9: Utilizarea șirurilor

1. Scrieți un șir de formatare care specifică o valoare șir aliniată la stânga, care trebuie să ocupe 24 de spații, urmată de o valoare de tip double aliniată la stânga, cu două cifre zecimale.

```
"%-24s  %-.2f"
```

2. Scrieți o secvență escape care reprezintă caracterul a cărui valoare ASCII este 45 în octal.

```
\045
```

3. Scrieți un apel de funcție și o atribuire care stochează în variabila \$r valoarea variabilei \$s și care elimină caracterele de tip spațiu alb de la început și de la sfârșit.

```
$r = trim($s);
```

4. Scrieți un apel de funcție care returnează un șir asemănător cu \$s, dar ale cărui n caractere, numărate de la poziția i , sunt înlocuite prin șirul \$r.

```
substr_replace($s, $r, $i, $n)
```

5. Scrieți o expresie regulată care corespunde numai sub-șirurilor "axb", "ayb" și "abz" care apar la sfârșitul unui șir subiect.

```
a[xyz]b$
```

Modulul 10: Utilizarea variabilelor cookie

1. Scrieți o instrucțiune PHP care creează o variabilă cookie denumită corect, care are valoarea "false"; stabiliți ca variabila cookie să expire în 30 de minute.

```
setcookie("corect", "false", time() + 1800);
```

2. Scrieți o instrucțiune PHP care șterge o variabilă cookie denumită trecut.

```
setcookie("trecut", "", time() - 3600);
```

3. Scrieți o instrucțiune PHP care afișează valoarea variabilei cookie denumite varsta.

```
echo "$varsta";
```

4. Scrieți o instrucțiune PHP care împachetează tabloul numit \$continut într-un șir denumit \$x.

```
$x = serialize($continut);
```

5. Scrieți o instrucțiune PHP care creează o variabilă cookie numită oriunde, care are valoarea "aici". Variabila cookie trebuie să expire în 30 de minute și trebuie să fie accesibilă în fiecare catalog al arborelui Web.

```
setcookie("oriunde", "aici", time() + 1800, "/");
```

Modulul 11: Lucrul cu fișiere și cataloage

1. Care este comanda UNIX care șterge catalogul (vid) test?

```
rmdir test
```

2. Care sunt privilegiile numerice pe care le veți atribui unui fișier pentru a acorda utilizatorului său numai accesul pentru citire și pentru a nu acorda altor utilizatori nici o categorie de acces?

```
0400 (octal)
```

3. Care este apelul de funcție care deschide fișierul test.txt, acordând accesul de atașare și de citire la un fișier?

```
fopen("test.txt", "a+")
```

4. Care este apelul de funcție care stabilește poziția pointerului fișierului asociat identificatorului \$f la sfârșitul fișierului?

```
fseek($f, 0, SEEK_END) sau fseek($f, filesize($f))
```

5. Care este apelul de funcție care returnează privilegiile asociate catalogului /test?

```
fileperms("/test");
```

Modulul 12: Expedierea și recepționarea mesajelor de poștă electronică

1. Care este protocolul folosit pentru expedierea mesajelor prin Internet?

SMTP

2. În ce mod contribuie funcțiile definite de utilizator la simplificarea activității de programare?

Funcțiile definite de utilizator vă permit să eliminați operațiile repetate și să le scrieți o singură dată. Astfel, un program devine mai scurt. De asemenea, aceste funcții vă permit să atribuiți un nume unei secvențe de operații.

3. Folosind funcția definită de utilizator adecvată descrisă în acest modul, scrieți o instrucțiune care copiază mesajul IMAP cu numărul 101 din dosarul curent în dosarul "test". Se presupune că variabila \$mb conține identificatorul asociat cu o conexiune IMAP deschisă, precum și că variabila \$pfx conține prefixul cutiei poștale IMAP.

```
copy_message($mb, $pfx, 101, "test");
```

4. Folosind funcția definită de utilizator adecvată descrisă în acest modul, scrieți o instrucțiune care modifică numele dosarului "test1" în "test2". Se presupune că variabila \$mb conține identificatorul asociat cu o conexiune IMAP deschisă, că variabila \$server conține șirul server IMAP (care include parantezele acolade, numele gazdei serverului, protocolul și numărul portului), că variabila \$pfx conține prefixul cutiei poștale IMAP, precum și că variabilele \$vechi, respectiv \$nou conțin numele dosarului.

```
rename_folder($mb, $server, $pfx, $vechi, $nou);
```

5. Folosind funcția definită de utilizator adecvată descrisă în acest modul, scrieți o instrucțiune care afișează antetele asociate mesajului IMAP al cărui număr este dat de valoarea variabilei \$n. Se va presupune că variabila \$mb conține identificatorul asociat cu o conexiune IMAP deschisă.

```
print_headers($mbx, $n);
sau
dump_headers($mbx, $n);
```

Modulul 13: Noțiuni fundamentale despre bazele de date și SQL

1. Cum se numește componenta unei baze de date relaționale care conține date referitoare la o instanță a unei entități?

Tabel

2. Cum se numește tipul de cheie care nu este, în general, unică pentru fiecare rând al unui tabel dintr-o bază de date?

Cheie externă

3. Care este cardinalitatea tipului de relație care trebuie eliminată în cursul procesului de modelare E-R?

N:N

4. Scrieți o comandă SQL care creează un tabel denumit test, care conține două câmpuri de câte 16 caractere fiecare, numite a și b.

```
CREATE TABLE test (a CHAR(15), b CHAR(15));
```

5. Scrieți o comandă SQL care inserează în baza de date creată la întrebarea anterioară un rând având ca valoare un șir de spații.

```
INSERT INTO test (a,b) VALUES (' ', ' ');
```

6. Scrieți o comandă SQL care raportează toate rândurile incluse în baza de date creată la întrebarea nr. 4.

```
SELECT * FROM test;
```

Modulul 14: Accesul la bazele de date relaționale

1. Scrieți un program PHP care se conectează la un server MySQL plasat la gazda numită db, folosind identificatorul de utilizator admin și parola secret.

```
$db = mysql_connect("db", "admin", "secret");
```

2. Scrieți un program PHP care selectează baza de date numită inventar în vederea unui acces ulterior.

```
mysql_select_db("inventar");
```

3. Scrieți un program PHP care execută interogarea stocată în variabila șir \$sql și stochează rezultatul în variabila \$rset.

```
$rset = mysql_query($sql);
```

4. Scrieți un program PHP care afișează numărul erorii asociate celei mai recente interogări MySQL.

```
echo mysql_errno();
```
5. Scrieți un program PHP care afișează valoarea primei coloane a rândului următor al setului de rezultate stocat în variabila \$rset.

```
$rand = mysql_fetch_row(); echo $rand[0];
```
6. Scrieți o buclă PHP care parcurge prin iterație rândurile unui set de rezultate, plasând fiecare rând în variabila \$rand. Bucla va fi configurată astfel încât să aibă un corp fără conținut.

```
while ($rand = mysql_fetch_row()) { }
```

Modulul 15: Utilizarea claselor și a obiectelor

1. Care este operatorul PHP folosit pentru instanțierea unui obiect?

```
new
```
2. Care este cuvântul cheie folosit pentru definirea unei clase?

```
class
```
3. Care este denumirea corectă a variabilelor incluse în cadrul unei clase?

```
Proprietăți
```
4. Care este denumirea corectă a funcțiilor incluse în cadrul unei clase?

```
Metode
```
5. Care este denumirea corectă a funcției speciale folosite la crearea unui obiect?

```
Constructor
```
6. Cum se mai numește o clasă părinte?

```
Clasă de bază
```
7. Cum se mai numește o clasă copil?

```
Clasă derivată
```
8. Cum se numește o metodă care este redefinită de o clasă copil?

```
Metodă anulată
```
9. Cum se numește o metodă care obține acces la valoarea unei proprietăți, dar nu o modifică?

```
Metodă accesoriu sau de obținere
```
10. Cum se numește o metodă care modifică valoarea unei proprietăți?

```
Metodă mutator sau de configurare
```

Modulul 16: Utilizarea șabloanelor de aplicație

1. Specificați două avantaje ale utilizării șabloanelor pentru organizarea unui sit Web de mari dimensiuni.
 Consecvența structurii și specializarea celui care desfășoară activitatea
2. Scrieți un bloc HTML care folosește o variabilă șablon numită legatura pentru a furniza adresa URL asociată unei legături. Textul asociat legăturii trebuie să fie „Duceți-vă acolo acum”.

```
<A HREF="{>Duceti-va acolo acum</A>
```
3. Scrieți o instrucțiune PHP care asociază valoarea 3.14159 cu variabila șablon pi a clasei FastTemplate. Se va presupune că variabila PHP \$t face referire la un obiect FastTemplate.

```
$t->assign('pi', '3.14159');
```
4. Scrieți o instrucțiune PHP care afișează valoarea asociată variabilei șablon html a clasei FastTemplate. Se va presupune că variabila PHP \$t face referire la un obiect FastTemplate.

```
$t->FastPrint('html');
```
5. Scrieți o instrucțiune PHP care instanțiază un obiect FastTemplate ce folosește șabloanele stocate în catalogul părinte al catalogului care conține scriptul PHP. Stocați referința la obiect într-o variabilă PHP numită \$ft.

```
$ft-> new FastTemplate('...');
```

Modulul 17: Depanarea scripturilor PHP

1. Care este numele tehnicii de depanare care implică și transformarea în comentarii a unor secțiuni de program?

```
Divide et impera
```
2. Include PHP 4 o funcționalitate de depanare?

```
Nu
```
3. Care este variabila PHP ce include textul celui mai recent mesaj de eroare, chiar dacă raportarea erorilor este suprimată?

```
$php_errormsg
```
4. Cum se numesc erorile care încalcă regulile gramaticale ale limbajului PHP?

```
Erori de sintaxă
```
5. Ce se întâmplă la apariția unei erori fatale, atunci când raportarea erorilor este suprimată?

```
Se încheie execuția programului
```

Anexa B:

Instalarea PHP

Această anexă descrie procedura de instalare a limbajului PHP și a programelor conexe, inclusiv a serverului Web Apache și a sistemului MySQL de gestiune a bazelor de date, în sistemele de operare frecvent folosite în calculatoarele de tip PC. Deoarece actualizările aduse limbajului PHP și programelor conexe pot influența procedura de instalare, trebuie să consultați informațiile existente la adresa <http://www.php.net> și, în măsura posibilităților, în situl Web al producătorului sistemului dumneavoastră de operare, pentru a vă pune la curent cu ultimele informații. Procedurile prezentate în această anexă sunt aplicabile numai sistemelor PC compatibile Intel; dacă doriți să instalați PHP pe un SPARC sau pe un alt sistem non-compatibil Intel, va trebui să urmați instrucțiunile de instalare specificate la <http://www.php.net> și în alte locații.

Instalarea și configurarea PHP și a programelor conexe poate depăși cu ușurință gradul de experiență al programatorilor versați, pentru a nu mai vorbi despre începători. O metodă alternativă simplă pentru a obține accesul la un server PHP este de a dobândi un cont la un furnizor de servicii Internet (ISP) care acceptă PHP. Situl Web PHP include o listă cu asemenea furnizori de servicii Internet (vezi adresa <http://www.php.net/links.php>).

Red Hat Linux 7.1

Red Hat Linux este o platformă extrem de populară pentru rularea PHP. Cea mai recentă versiune a sistemului Red Hat Linux este organizată astfel încât să faciliteze instalarea PHP și a programelor conexe. Sub-secțiunile următoare conțin instrucțiuni pentru instalarea pachetelor RPM referitoare la PHP sub Red Hat Linux. În funcție de configurația sistemului dumneavoastră, poate fi necesar să instalați pachete suplimentare, pentru satisfacerea dependențelor asociate pachetelor menționate în comenzi.

Instalarea serverului Apache

Discul 1 de instalare a sistemului de operare Red Hat Linux 7.1 conține pachetul RPM pentru serverul Web Apache. Puteți instala acest pachet prin emiterea următoarelor comenzi:

```
su -
mount -t iso9660 /dev/cdrom /mnt/cdrom
cd /mnt/cdrom/RedHat/RPMS
rpm --replacepkgs -Uvh apache-1*.rpm
cd
```

```
umount /mnt/cdrom
exit
```

Instalarea PHP

Discul 1 de instalare a sistemului de operare Red Hat Linux conține pachetul RPM pentru PHP. Puteți instala acest pachet prin emiterea următoarelor comenzi:

```
su -
mount -t iso9660 /dev/cdrom /mnt/cdrom
cd /mnt/cdrom/RedHat/RPMS
rpm --replacepkgs -Uvh php-4*.rpm
cd
umount /mnt/cdrom
exit
```

Instalarea sistemului MySQL

Discul 2 de instalare a sistemului de operare Red Hat Linux conține pachetele RPM pentru sistemul MySQL de gestiune a bazelor de date. Puteți instala aceste pachete prin emiterea următoarelor comenzi:

```
su -
mount -t iso9660 /dev/cdrom /mnt/cdrom
cd /mnt/cdrom/RedHat/RPMS
rpm --replacepkgs -Uvh \
  mysql-3*.rpm \
  mysql-server-*.rpm \
  mysqlclient9-*.rpm \
  php-mysql-*.rpm
cd
umount /mnt/cdrom
exit
```

Instalarea IMAP

Discul 2 de instalare a sistemului de operare Red Hat Linux conține pachetele RPM pentru instalarea sistemului de poștă IMAP. Puteți instala aceste pachete prin emiterea următoarelor comenzi:

```
su -
mount -t iso9660 /dev/cdrom /mnt/cdrom
cd /mnt/cdrom/RedHat/RPMS
rpm --replacepkgs -Uvh \
  imap-2000-9*.rpm \
  php-imap-*.rpm
cd
umount /mnt/cdrom
exit
```


Configurarea sistemului MySQL

Din păcate, configurarea sistemului MySQL sub Red Hat Linux 7.1 este oarecum greoaie. Pare probabil că această problemă va fi rezolvată în curând de Red Hat. În momentul scrierii rândurilor de față, configurarea sistemului MySQL era posibilă prin emiterea următoarelor comenzi:

```
su -
service mysqld start
su mysql
mysql_install_db
mysqladmin -u root password 'parola'
mysqladmin -p -u root -h nume password 'parola'
mysqladmin -p -u root -h gazda password 'parola'
mysqladmin -p -u root -h localhost password 'parola'
mysqladmin -p -u root -h localhost.localdomain password 'parola'
exit
exit
```

unde:

- *parola* este parola care urmează a fi asociată utilizatorului rădăcină
- *nume* este numele de domeniu complet determinat al gazdei locale
- *gazda* este numele gazdei locale

După ce a fost emisă prima comandă `mysqladmin`, celelalte comenzi `mysqladmin` solicită parola stabilită la prima comandă. După emiterea acestei comenzi, utilizatorul rădăcină poate folosi programele `mysql` și `mysqladmin`. În particular, utilizatorul rădăcină poate folosi programul `mysqladmin` pentru a crea noi utilizatori.

Pornirea serviciului Apache

După instalarea PHP și a programelor conexe, trebuie pornite serviciile asociate. Serviciul MySQL a fost pornit de scriptul de configurare dat anterior. Totuși, serverul Web Apache trebuie pornit manual. Pentru aceasta, emiteți următoarele comenzi:

```
su -
service httpd start
exit
```

Acest server trebuie pornit din nou la fiecare modificare a fișierului de configurație `Httpd`, în speță `/etc/httpd.conf`. Pentru a porni din nou serverul, emiteți următoarele comenzi:

```
su -
service httpd stop
service httpd start
exit
```

Alternativ, puteți porni din nou sistemul; Apache va porni automat atunci când sistemul rulează la nivelul 3 sau la un nivel superior.

Testarea instalării

Pentru a verifica dacă Apache și PHP sunt instalate și rulează, mai întâi orientați un browser spre gazda unde sunt instalate acestea și verificați dacă browserul poate „vedea” pagina de test Apache. De exemplu, folosind un browser care rulează chiar pe gazdă, orientați browserul spre `http://localhost/`. Browserul Lynx este adecvat pentru acest scop, deoarece nu necesită o interfață grafică cu utilizatorul funcțională. Pentru a utiliza Lynx ca să verificați dacă Apache funcționează, emiteți comanda:

```
lynx http://localhost/
```

Pentru a verifica dacă PHP funcționează corect, creați următorul script, plasându-l în fișierul `/var/www/html/phpinfo.php`:

```
<?php
phpinfo();
?>
```

Apoi, orientați un browser spre adresa URL asociată scriptului, adică `http://gazda/phpinfo.php`, unde *gazda* este numele gazdei pe care rulează serverul Apache. Dacă browserul rulează pe aceeași gazdă ca și Apache, puteți specifica *localhost* ca valoare a variabilei *gazda*. Dacă PHP rulează, veți vedea ecranul de informații PHP asociat cu funcția `phpinfo()`.

În acest moment, puteți modifica opțiunile de configurare PHP. Pentru aceasta, editați fișierul `/etc/php.ini` conform necesităților. Apoi, reporniți serverul Apache prin emiterea comenzilor:

```
su -
service httpd stop
service httpd start
exit
```

Alte versiuni de Linux și UNIX

În general, instalarea PHP și a programelor conexe sub Red Hat Linux 6.2 și alte versiuni de Linux și UNIX impun construirea PHP – și probabil a unora sau a tuturor programelor conexe – pornind de la codul sursă. Similar, instalarea unei versiuni actualizate a PHP sau a programelor conexe care nu au fost încă împachetate și distribuite de Red Hat va impune, în general, construirea PHP pornind de la codul sursă.

Notele și sugestiile asociate procedurii de instalare și configurare pentru PHP sunt documentate în manualul PHP pe suport electronic, disponibil la adresa `http://www.php.net/manual/en/installation.php`. Construirea PHP pornind de la codul sursă impune următoarele operații:

- Descărcarea fișierelor sursă pentru PHP și Apache

- Descărcarea fișierelor sursă pentru toate programele conexe pe care doriți să le utilizați, precum MySQL sau IMAP
- Instalarea instrumentelor necesare de dezvoltare a programelor, inclusiv un compilator C, utilitarul make, flex, bison și alte instrumente necesare pentru compilarea fișierelor sursă
- Configurarea și compilarea programelor corelate sub formă de biblioteci accesibile pentru Apache/PHP
- Configurarea și compilarea PHP
- Configurarea, compilarea și instalarea serverului Apache

Etapele pe care trebuie să le parcurgeți pentru a executa aceste operații se modifică în mod constant, o dată cu lansarea de noi versiuni ale limbajului PHP, serverului Apache sau programelor conexe. În consecință, pentru a avea șanse logice de succes, trebuie să respectați instrucțiunile asociate fiecărei versiuni de program. Acest lucru este îngreunat de faptul că versiunile PHP, Apache și versiunile programelor conexe nu sunt sincronizate. Așadar, actualizările unei aplicații pot impune modificări în procedura de instalare a altor aplicații. Procedura de instalare revizuită, totuși, s-ar putea să nu fie disponibilă decât după lansarea unei noi versiuni a aplicației afectate. Mai mult, instrucțiunile de instalare presupun, în general, o cunoaștere aprofundată a comenzilor UNIX/Linux, precum și a administrării sistemelor.

Ca atare, pentru a instala și configura PHP, trebuie să consultați resursele menționate în Anexa C. În particular, veți descoperi că grupurile de discuții și listele de corespondență reprezintă o importantă sursă de asistență la instalarea și configurarea limbajului PHP.

Windows NT/2000 și 95/98

Pentru a instala PHP sub un sistem de operare Windows, mai întâi trebuie să instalați și să configurați un server Web acceptat. Sub Windows NT/2000, puteți folosi:

- Apache
- Microsoft Internet Information Server (IIS), versiunea 4.0 sau ulterioară
- Netscape Enterprise Server sau iPlanetServer
- O'Reilly Website Pro
- Xitami

Sub Windows 95/98, puteți folosi:

- Apache
- Microsoft Personal Web Server (PWS), recomandat numai pentru Windows 98
- O'Reilly Website Pro
- Xitami

Instrucțiunile următoare explică procedura de instalare a serverelor Apache, IIS și PWS. Pentru informații referitoare la utilizarea unui alt server Web, cum ar fi O'Reilly Website Pro, consultați documentația aferentă serverului Web respectiv.

Apache

Pentru a instala Apache, consultați distribuția binară Windows de la adresa <http://www.apache.org> și urmați instrucțiunile de instalare și configurare date la adresa <http://httpd.apache.org/docs/windows.html>.

IIS

Pentru a instala IIS sub Windows NT, descărcați Windows NT 4.0 Option Pack, disponibil prin intermediul paginii Web NT Server de la adresa <http://www.microsoft.com/ntserver/>. Microsoft își reorganizează sistematic situl Web propriu, deci este posibil să aveți nevoie de funcționalitatea de căutare a sitului pentru a localiza fișierul. În momentul scrierii acestor rânduri, programul respectiv era accesibil la adresa <http://www.microsoft.com/ntserver/nts/downloads/recommended/NT40ptPk/default.asp>.

Microsoft IIS este inclus în distribuția Windows 2000. Pentru a instala IIS sub Windows 2000, selectați Start | Settings | Control Panel | Add/Remove Programs | Add/Remove Windows Components. În cazul în care caseta de validare IIS este activată, înseamnă că IIS este deja instalat. În caz contrar, activați caseta de validare, executați clic pe Next și respectați instrucțiunile de pe ecran.

PWS

Pentru a instala PWS sub Windows 98, introduceți compact discul de distribuție Windows 98 în unitatea CD-ROM a sistemului dumneavoastră. Folosind Windows Explorer, treceți la catalogul cu module add-on și apoi în sub-catalogul pws al acestuia. Executați dublu clic pe fișierul setup.exe care se găsește acolo și urmați instrucțiunile afișate pe ecran pentru a instala PWS. Rețineți catalogul pe care l-ați selectat drept catalog de bază prestabilit pentru publicarea paginilor Web. Prin convenție, acesta este catalogul C:\inetpub\wwwroot; cu toate acestea, puteți selecta și un alt catalog, dacă preferați.

PHP

Pentru a instala PHP, descărcați versiunea binară CGI Win32 a limbajului PHP de la adresa <http://www.php.net>. Apoi, decompriți arhiva care conține fișierul de distribuție și urmați instrucțiunile date în fișierul install.txt.

Anexa C:

Resurse PHP

Această anexă identifică unele resurse care vă pot fi de ajutor la instalarea, configurarea și utilizarea limbajului PHP și a programelor sale conexe.

Situri Web

- PHP Conference Material, de la adresa conf.php.net: conține prezentări susținute la diferite conferințe pe teme de PHP
- PHP Web Ring, de la adresa nav.webring.yahoo.com/hub?ring=php&list, un inel Web care conține aproximativ 75 de situri cu materiale referitoare la PHP
- PHP Resource Index (indice de resurse PHP), de la adresa php.resourceindex.com: programe PHP, documentație și comunități
- PHP Classes Repository (depozit de clase PHP), de la adresa phpclasses.upperdesign.com: clase PHP disponibile gratuit
- PX The PHP Code Exchange (bursa de programe PHP), de la adresa px.sklar.com: programe PHP
- HotScripts.com Web Development Portal (portalul de dezvoltare Web HotScripts.com), de la adresa www.hotscripts.com/PHP: liste de cărți despre PHP, articole despre PHP, comunități, programe, instrumente, sugestii, manuale și situri Web
- PHP Editors, de la adresa www.itworks.demon.co.uk/phpeditors.htm: o listă a editoarelor compatibile PHP
- Linux Guruz, de la adresa www.linuxguruz.org: un sit care furnizează numeroase legături cu resurse referitoare la PHP și Linux
- Situl Web de bază al sistemului MySQL, www.mysql.com
- Situl Web de bază al limbajului PHP, www.php.net
- PHPBuilder, de la adresa www.phpbuilder.com: articole, manuale, forumuri Web și programe referitoare la PHP
- phpWizard.net, de la adresa www.phpwizard.net: articole, manuale și proiecte pentru dezvoltatorii Web
- WeberDev.com, www.weberdev.com: exemple de programe, articole și manuale pentru dezvoltatorii Web
- Zend.com, de la adresa www.zend.com, un dezvoltator de tehnologie PHP

Liste de corespondență

PHP folosește numeroase liste de corespondență sponsorizate. Aceste liste sunt arhivate la adresele www.php.net și www.phpbuilder.com, care mai conțin și pagini Web pentru abonarea la aceste liste:

- [php-db](#), o listă de corespondență pentru utilizatorii de baze de date cu PHP
- [php-developer-list](#), o listă de corespondență pentru dezvoltatorii PHP
- [php-general](#), o listă de corespondență pe subiecte generale legate de PHP
- [php-i18n](#), o listă de corespondență pentru proiectul de internaționalizare PHP
- [php-install](#), o listă de corespondență pentru asistență în instalarea limbajului PHP
- [phplib-dev-list](#), o listă de corespondență pentru dezvoltatorii PHPLib
- [phplib-list](#), o listă de corespondență pentru utilizatorii PHPLib
- [php-migration](#), o listă de corespondență pentru cei care migrează sisteme la (sau de la) PHP
- [php-windows](#), o listă de corespondență pentru utilizatorii de PHP sub Microsoft Windows

Grupuri de informare

Numeroase grupuri de informare USENET abordează subiecte legate de PHP. Dacă furnizorul dumneavoastră de servicii Internet nu permite accesul la grupurile de informare, puteți citi știrile publicate la adresa groups.google.com. Grupurile de informare relevante sunt următoarele:

- [php-dev](#), un grup de informare pentru dezvoltatorii PHP
- [php.general](#), un grup de informare pentru utilizatorii PHP
- [php.windows](#), un grup de informare pentru utilizatorii PHP sub Microsoft Windows

Anexa D:

Elemente fundamentale ale sistemului de operare UNIX

În această anexă sunt explicate comenzile UNIX elementare acceptate de majoritatea sistemelor de operare asemănătoare cu UNIX, inclusiv Linux. Materialul de față se concentrează asupra comenzilor simple, de genul celor utilizate în timpul lucrului sub un cont pe o gazdă UNIX, cum sunt cele asigurate de numeroși furnizori de servicii Internet.

Pentru a administra un sistem UNIX, nu doar pentru a-l utiliza, va trebui să cunoașteți un alt set de comenzi, ceva mai complex. Pentru a învăța mai multe despre administrarea sistemelor UNIX, consultați volumul *UNIX: The Complete Reference*, de Kenneth Rosen și Doug Host (Osborne McGrawHill, 1999).

Prima secțiune a acestei anexe se referă la termenii și conceptele elementare, importante pentru înțelegerea atât a sistemului de operare UNIX, cât și a comenzilor aferente acestuia. Cea de-a doua secțiune a anexei explică modul de efectuare a operațiilor comune din UNIX.

Concepte UNIX elementare

Ca și alte sisteme de operare cunoscute, UNIX este construit pe baza unui set de concepte fundamentale. Cele mai importante din aceste concepte sunt următoarele:

- Fișierele, care stochează date
- Cataloagele, care conțin fișiere și alte cataloage (numite sub-cataloage)
- Căile, care descriu locațiile fișierelor și ale cataloagelor

Deoarece UNIX este un sistem de operare multiutilizator, numeroase alte concepte elementare sunt de asemenea relevante. Cele mai importante din acestea sunt următoarele:

- Conturile de utilizator, care sunt asociate cu utilizatorii autorizați ai sistemului
- Grupurile de utilizatori, care reprezintă seturi formate din unul sau mai mulți utilizatori care pot fi tratați ca o unitate
- Permisunile, care controlează grupurile și utilizatorii ce au permisiunea de a obține accesul la un catalog sau la un fișier specificat

Fișiere

Similar altor sisteme de operare cunoscute, UNIX stochează datele în entități cunoscute sub numele de *fișiere*. Fișierele UNIX pot conține text, date ASCII sau date binare, cum sunt programele executabile.

Numele pe care îl atribuți unui fișier UNIX trebuie să satisfacă un set de reguli, care sunt oarecum diferite de la o varietate UNIX la alta. Pentru a fi feriți de pericole, puteți respecta aceste reguli extrem de conservatoare:

- Lungimea numelui nu trebuie să depășească 32 de caractere
- Numele trebuie să înceapă cu un caracter scris cu minusculă
- Numele trebuie să conțină numai litere scrise cu minuscule, cifre și caractere de subliniere; opțional, numele fișierului poate conține un singur punct, după care nu trebuie să urmeze mai mult de 3 caractere

Regulile efective impuse de UNIX pentru denumirea fișierelor sunt cu mult mai puțin restrictive decât acestea. Totuși, dacă respectați regulile prezentate anterior, veți evita problemele care pot apărea la deplasarea fișierelor dintr-un sistem UNIX într-un sistem care nu folosește UNIX, respectiv la deplasarea în sens invers. De exemplu, numele fișierelor Microsoft Windows nu sunt sensibile la diferența între majuscule și minuscule, dar numele fișierelor UNIX prezintă această sensibilitate. Cu alte cuvinte, în Microsoft Windows numele *abc* și *ABC* reprezintă același fișier, dar în UNIX reprezintă referințe la fișiere distincte. Folosind numai minuscule în numele fișierelor UNIX, evitați confuziile care pot apărea la mutarea fișierelor de la un sistem de operare la altul.

Cataloage

Cataloagele, cunoscute uneori și sub denumirea de *dosare*, conțin fișiere și alte cataloage. Un catalog inclus într-un alt catalog se numește *sub-catalog*. Catalogul care conține un catalog dat se numește *catalogul părinte* al catalogului dat.

Fiecare sistem are un catalog special, numit *catalog rădăcină*, reprezentat prin intermediul unui singur caracter slash orientat înainte (/). Catalogul rădăcină are sub-cataloage care, la rândul lor, conțin alte sub-cataloage. Aceste sub-cataloage formează o ierarhie în vârful căreia se găsește catalogul rădăcină. Rezultatul este similar celui prezentat în figura D-1.

Căi

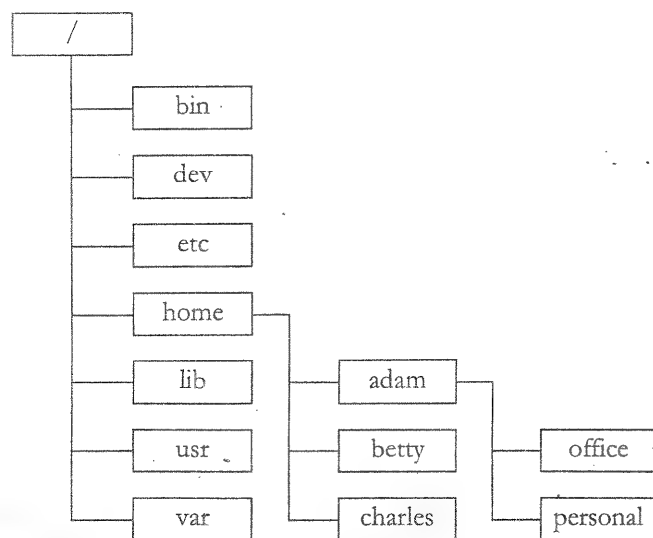
Numele cataloagelor nu trebuie să fie unice într-un anumit sistem. În caz contrar, doi utilizatori – să-i numim Adam și Betty – nu vor putea să creeze amândoi cataloage numite *personal*. Numele complet al unui catalog este dat de *calea sa*, adică de succesiunea de cataloage parcurse de la catalogul rădăcină pentru a se ajunge la

catalogul respectiv. Fiecare componentă a unei căi este separată de vecina sa prin intermediul unui caracter slash orientat înainte, același simbol fiind folosit și pentru desemnarea catalogului rădăcină. Practic, acest caracter de separare poate fi considerat ca echivalent comenzii „deplasare la *catalog*“, unde *catalog* este catalogul specificat urmând „traseul“ indicat de caracterul respectiv. Pentru a urma o cale, începeți de la catalogul rădăcină și traversați fiecare catalog specificat, până când ajungeți la sfârșitul căii.

De exemplu, în figura D-1 utilizatorii Adam și Betty au fiecare câte un catalog. Catalogul personal al lui Adam are calea `/home/adam/personal`. Dacă Betty ar fi avut un catalog personal, calea sa ar fi fost `/home/betty/personal`. Uneori, un caracter slash orientat înainte `/` este adăugat la sfârșitul unei căi, ceea ce nu afectează semnificația căii. Căile `/home/adam/personal` și `/home/adam/personal/` sunt considerate ca fiind una și aceeași entitate.

Uneori, este convenabilă abrevierea unei căi, exprimând-o în raport cu un catalog, altul decât catalogul rădăcină. O asemenea cale se numește *cale relativă*, iar catalogul relativ la care este dată o cale relativă se numește *catalogul de bază* al căii; o cale exprimată relativ la catalogul rădăcină se numește *cale absolută*. Căile absolute încep întotdeauna cu un caracter slash orientat înainte, pentru a indica asocierea acestora la catalogul rădăcină. Căile relative încep întotdeauna cu numele unui catalog; mai concret, cu numele primului catalog care trebuie parcurs după catalogul de bază.

Relativ la catalogul `/home`, catalogul personal al lui Adam are calea `/adam/personal`. Relativ la catalogul `/home/adam`, catalogul personal al lui Adam are calea `personal`.



Figură D-1 O ierarhie de cataloage simplificată.

Pentru exprimarea căilor relative se folosesc, uneori, două simboluri speciale. Simbolul `..` simbolizează catalogul părinte, iar simbolul `.` precizează catalogul curent. Relativ la catalogul `/home/betty`, catalogul personal al lui Adam are calea `../adam/personal`. Relativ la catalogul `bin`, catalogul personal al lui Adam are calea `../home/adam/personal`.

Și fișierele au căi. Calea unui fișier este o cale absolută sau relativă asociată catalogului care conține fișierul, urmată de un caracter slash orientat înainte `/`, urmat de numele fișierului. De exemplu, fișierul `index.html` din catalogul `/var/www/html` poate fi identificat prin calea `/var/www/html/index.html`. Numele unui fișier este el însuși o categorie specială de cale relativă, în speță o cale relativă la catalogul care conține fișierul.

Conturi de utilizator

Pentru a controla accesul la un sistem UNIX, utilizatorii autorizați primesc *conturi de utilizator*. În mod obișnuit, la un cont de utilizator sunt asociate numeroase caracteristici, precum următoarele:

- Numele utilizatorului
- Un nume de utilizator special, folosit pentru identificarea utilizatorului
- Un identificator numeric al utilizatorului
- O parolă folosită pentru confirmarea identității utilizatorului

În general, unui utilizator îi este atribuit un catalog special, pentru uz personal. Acest catalog se numește *catalog de bază* al utilizatorului.

Administratorul unui sistem UNIX folosește un cont de utilizator special, la care este asociat numele de utilizator *root* (rădăcină). Acest administrator de sistem poate efectua operații care sunt interzise utilizatorilor obișnuiți.

Grupuri de utilizatori

Uneori, este convenabil să se poată face referire la un set de conturi de utilizator. În acest scop, UNIX permite administratorului de sistem să definească *grupuri de utilizatori*. Fiecare cont UNIX face parte dintr-unul sau mai multe grupuri de utilizatori.

Proprietate și permisiuni

Fiecare fișier UNIX are un set de *permisiuni* asociate, care determină operațiile pe care utilizatorii au autorizația să le execute. Când un utilizator creează un fișier sau un catalog, utilizatorul îi poate atribui acestuia permisiuni. Administratorul de sistem poate proceda într-un mod similar. Permisunile posibile sunt:

- Read (citire), care permite unui utilizator să vizualizeze conținutul unui fișier

- Write (scriere), care permite unui utilizator să trunchieze conținutul unui fișier, să aducă modificări în conținutul unui fișier sau să atașeze date noi la sfârșitul fișierului
- Execute (execuție), care permite unui utilizator să ruleze un script sau un program binar executabil

Similar, fiecare catalog UNIX are un set de permisiuni asociate. Aceste permisiuni au aceleași nume ca și permisiunile asociate fișierelor, dar au semnificații oarecum diferite atunci când sunt aplicate unui catalog:

- Read (citire), care permite unui utilizator să determine fișierele și sub-cataloagele incluse în catalog
- Write (scriere), care permite unui utilizator să creeze fișiere și sub-cataloage noi în cadrul catalogului, precum și să șteargă fișiere și cataloage din cadrul catalogului
- Execute (execuție), care permite unui utilizator să obțină acces la fișiere și sub-cataloage din cadrul catalogului

Notă

Pentru a șterge un fișier, un utilizator are nevoie numai de acces de scriere la catalogul care conține fișierul. Utilizatorul nu are nevoie de acces de scriere la fișierul în sine.

Fiecare fișier sau catalog are asociat un cont de utilizator, cunoscut sub numele de *posesorul* fișierului sau al catalogului respectiv, precum și un grup de utilizatori, cunoscut sub numele de *grup posesor*. Unele sisteme UNIX permit posesorului unui fișier sau al unui catalog să transfere dreptul de proprietate unui alt cont de utilizator. Toate sistemele UNIX permit posesorului unui fișier sau al unui catalog să transfere proprietatea de grup a fișierului sau a catalogului către orice grup din care face parte posesorul.

Posesorul unui fișier sau al unui catalog poate primi una sau mai multe permisiuni posibile (citire, scriere și execuție), prin care sunt specificate operațiile pe care posesorul are dreptul să le execute. Similar, grupul posesor al unui fișier poate primi una sau mai multe permisiuni posibile. În final, una sau mai multe permisiuni sunt asociate *altor persoane*, adică unor utilizatori care nu sunt nici posesori ai fișierului sau ai catalogului și nici nu fac parte din grupul posesor.

De exemplu, un fișier poate primi următoarele permisiuni:

- Posesor: citire, scriere
- Grup posesor: citire
- Alte persoane: nici una

Aceste permisiuni autorizează posesorul să citească și să scrie, dar nu să execute fișierul. Permisuniile respective autorizează membrii grupului posesor al fișierului să citească, dar nu să scrie în fișier sau să execute fișierul. Ceilalți utilizatori – care nu

sunt nici posesori ai fișierului și nici nu fac parte din grupul posesor al fișierului – nu au nici un fel de permisiuni de acces la fișier.

Tehnici UNIX elementare

Această secțiune explică modul de utilizare a comenzilor UNIX pentru efectuarea operațiilor frecvent utilizate. UNIX este un sistem complex și de mari dimensiuni, deci informațiile nu sunt detaliate. Pentru mai multe informații cu privire la comenzile UNIX menționate în această secțiune, precum și la alte comenzi UNIX pe care le puteți utiliza, consultați referințele specificate la începutul prezentei anexe.

UNIX poate fi accesibil prin intermediul unei interfețe bazate pe text, numită în linie de comandă sau utilizând o interfață grafică cu utilizatorul. Majoritatea utilizatorilor care obțin acces la un sistem UNIX prin Internet folosesc interfața în linie de comandă. Din acest motiv și deoarece UNIX acceptă mai multe interfețe grafice cu utilizatorul, care prezintă caracteristici de operare oarecum diferite unele în raport cu altele, această secțiune va trata numai despre linia de comandă UNIX.

De asemenea, sistemele UNIX acceptă o varietate de interfețe în linie de comandă, cunoscute sub numele de *shell*. Cele mai populare interfețe shell sunt interfața Bourne și derivatele sale, precum interfața shell de tip Bourne-again (BASH), frecvent configurată în mod prestabilit pe sistemele Linux. Dacă sistemul dumneavoastră UNIX este configurat astfel încât să folosească o altă interfață shell decât o interfață de tip Bourne, comenzile prezentate în această secțiune pot avea o funcționare diferită de cea prezentată.

În acest caz, puteți solicita administratorului de sistem să vă configureze contul astfel încât să folosească o interfață Bourne, cum este interfața shell BASH. În general, aceasta este o cerere simplă, pe care administratorul de sistem o poate satisface în câteva secunde.

Deschiderea și închiderea sesiunii de lucru

În general, puteți deschide sesiunea de lucru cu un sistem UNIX de la consola sistemului sau puteți folosi de la distanță un program precum Telnet sau SSH. Cu excepția stațiilor de lucru personale, majoritatea utilizatorilor obțin de la distanță accesul la sisteme. Protocolul SSH este mai sigur decât protocolul Telnet, care trimite nume de utilizator și parole fără a le cripta. Deci, dacă sistemul la care doriți să obțineți accesul acceptă SSH, trebuie să folosiți SSH în locul protocolului Telnet, pentru a reduce riscul de compromitere a contului dumneavoastră.

În general, sistemele Microsoft Windows și UNIX furnizează clienți Telnet ca parte a configurației standard. Pentru a obține accesul la o gazdă aflată la distanță prin intermediul protocolului Telnet, emiteți comanda:

`telnet gazda` ← *gazda este numele sau adresa IP a gazdei la care doriți să obțineți accesul*

Ca răspuns la comandă, gazda aflată la distanță vă va solicita numele de utilizator și parola dumneavoastră. Parola nu va fi afișată pe ecran atunci când o tastați, pentru a evita dezvăluirea acesteia posibilor privitori nedorți.

În general, sistemele UNIX furnizează un client SSH ca parte a configurației standard. În funcție de configurația protocolului SSH existentă în serverul dumneavoastră, poate că este necesar să pre-configurați contul dumneavoastră de utilizator pentru acces prin intermediul SSH. Totuși, în general este permis accesul prin SSH. În acest caz, puteți obține accesul la o gazdă aflată la distanță prin intermediul SSH, prin emiterea următoarei comenzi:

`ssh -l nume_utilizator gazda` ← *nume_utilizator este numele de utilizator care v-a fost repartizat, iar gazda este numele sau adresa IP a gazdei la care doriți să obțineți accesul*

Sistemul vă poate cere parola, dar nu o va afișa pe ecran atunci când o tastați.

Sistemele Microsoft Windows nu furnizează un client SSH ca parte a configurației standard. Între programele client SSH frecvent folosite sub Windows se numără și programul gratuit *putty*, care poate fi obținut de la adresa <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

Când deschideți sesiunea de lucru la un sistem UNIX – prin intermediul unei console, al protocolului Telnet sau al unui client SSH – sistemul UNIX vă prezintă o ilustrație de conectare și, în cele din urmă, un prompt de comandă. În mod caracteristic, acesta din urmă este un șir de caractere care se încheie cu un simbol al dolarului. De exemplu, puteți vedea

```
[mccarty@athlon mccarty]$
```

Promptul de comandă vă arată că sistemul este gata de a primi comenzile dumneavoastră.

Închiderea sesiunii de lucru cu un sistem UNIX este oarecum mai simplă decât deschiderea sesiunii de lucru. Pur și simplu tastați comanda `exit` și apăsați pe tasta ENTER. Veți primi un mesaj care confirmă reușita închiderii procesului de lucru.

Emiterea unei comenzi UNIX

Pentru a emite o comandă UNIX, tastați comanda după promptul de comandă și apoi apăsați pe tasta ENTER. Multe sisteme UNIX pun la dispoziție taste speciale, care vă permit să editați comanda curentă și să repetați comenzile anterioare. Exersați utilizarea tastelor cu săgeți, de exemplu, pentru a vedea care sunt posibilitățile de care dispuneți.

Când o comandă UNIX este prea lungă pentru a se încadra într-o singură linie, puteți continua linia tastând un caracter slash orientat înapoi \ ca ultim caracter al liniei, după care apăsați pe tasta ENTER.

În general, comenzile UNIX includ trei părți, din care ultimele două sunt opționale:

- Numele comenzii
- Opțiunile comenzii, care sunt litere sau cuvinte precedate de o liniuță (-)
- Argumente, care pot fi nume de fișiere, nume de cataloage sau texte arbitrare

De exemplu, iată o comandă UNIX caracteristică:

```
chown -R mccartyb /home/mccartyb
```

Modificarea parolei

Pentru a vă modifica parola UNIX, emiteți comanda `passwd`, care nu are nevoie de opțiuni sau argumente. Comanda vă va solicita parola dumneavoastră curentă și vă va cere de două ori noua parolă:

```
[mccarty@athlon mccartyb]$ passwd
Changing password for mccartyb
(current) Unix password:
New Unix password:
Retype new Unix password:
passwd: all authentication tokens updated successfully
[mccarty@athlon mccartyb]$
```

Observați că informațiile referitoare la parolă nu sunt afișate pe ecran.

Determinarea utilizatorilor conectați

Pentru a vedea care sunt utilizatorii conectați la o gazdă UNIX, emiteți comanda `w`. Comanda va raporta o varietate de informații referitoare la starea sistemului și la utilizatorii conectați în momentul respectiv:

```
[mccarty@athlon mccartyb]$ w
11:46 am up 21 days, 22:05, 2 users, load average: 0.00, 0.00, 0.00
USER  TTY      FROM             LOGIN@      IDLE   JCPU   PCPU   WHAT
root  tty1          17May01         7days  0.23s  0.23s  bash
root  pts/0  192.168.0.3      11:44am    0.00s  0.08s  0.02s  w
[mccarty@athlon mccartyb]$
```

De exemplu, datele de ieșire prezentate anterior arată că administratorul de sistem a deschis sesiunea de lucru de două ori. O sesiune a fost inițiată la 17 mai, iar cealaltă la ora 11,44 a.m. astăzi.

Verificarea stării de activitate a unei gazde

Pentru a verifica dacă o gazdă este activă, emiteți comanda

```
ping -c 3 gazda ← gazda este numele sau adresa IP a gazdei
```

În cazul în care facilitățile de rețea ale gazdei sunt operaționale, comanda va raporta intervalul de timp necesar unui singur pachet pentru a ajunge la gazdă. Acest interval de timp se măsoară de trei ori, după care se va prezenta un raport de sumar. De exemplu:

```
[mccartyb@athlon mccartyb]$ ping -c 3 www.osborne.com
PING www.osborne.com (198.45.24.130) from 192.168.0.12 : 56(84) bytes of data.
64 bytes from 198.45.24.130: icmp_seq=0 ttl=241 time=106.986 msec
64 bytes from 198.45.24.130: icmp_seq=1 ttl=241 time=79.953 msec
64 bytes from 198.45.24.130: icmp_seq=2 ttl=241 time=79.962 msec

--- www.osborne.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 79.963/88.967/106.986/12.741 ms
[mccartyb@athlon mccartyb]$
```

Citirea și expedierea mesajelor de e-mail

UNIX acceptă o varietate de clienți pentru expedierea și recepționarea mesajelor de e-mail. Mulți administratori de sistem UNIX instalează popularul program de e-mail *pine*. Pentru a lansa acest program, emiteți comanda

```
pine
```

Programul client de e-mail *pine* include documentație încorporată, concepută pentru a vă ajuta să învățați modul de utilizare a acestuia. Programul este controlat prin meniuri; puteți folosi tastele de deplasare a cursorului pentru a evidenția un articol de meniu, după care alegeți articolul apăsând pe tasta ENTER. Pe lângă expedierea și recepționarea mesajelor de e-mail, programul *pine* poate citi și publica mesaje pentru grupurile de informare.

Majoritatea sistemelor UNIX acceptă mai puțin sofisticata comandă *mail*, care este adecvată pentru necesitățile elementare ale utilizatorilor poștei electronice. Pentru a învăța să utilizați comanda *mail*, citiți pagina de manual aferentă comenzii, așa cum este explicat în secțiunea „Consultarea documentației UNIX pe suport electronic”.

Modificarea catalogului de lucru curent

Când un utilizator deschide sesiunea de lucru cu un sistem UNIX, catalogul de bază al utilizatorului este desemnat ca fiind *catalog de lucru curent*, respectiv *catalog curent*. Catalogul curent este asimilat catalogului de bază pentru orice cale relativă

specificată într-o comandă UNIX. Un utilizator poate modifica identitatea catalogelor curente de lucru prin emiterea comenzii

```
cd cat
```

unde *cat*, catalogul care devine catalog curent, este specificat printr-o cale absolută sau relativă. De exemplu, un utilizator poate transforma */bin* în catalog curent prin emiterea comenzii

```
cd /bin
```

Majoritatea sistemelor UNIX sunt configurate astfel încât să prezinte numele catalogului curent ca parte a promptului de comandă. De exemplu, promptul de comandă

```
[mccartyb@athlon bin]$
```

arată că utilizatorul a deschis sesiunea de lucru sub contul *mccartyb* la o gazdă numită *athlon*, precum și că *bin* este numele catalogului curent.

Puteți emite comanda

```
pwd
```

pentru a determina catalogul curent. Comanda afișează calea absolută a catalogului curent.

Vizualizarea conținutului unui catalog

Pentru a vedea o listă a fișierelor și a sub-catalogelor incluse în catalogul curent, emiteți comanda

```
ls
```

Comanda afișează mai multe fișiere sau cataloage pe fiecare linie, astfel:

```
index.html  php  php.tgz
```

Unele sisteme UNIX vor face diferența între fișiere și cataloage prin afișarea numerelor acestora în culori distincte. Dacă doriți să vizualizați conținutul unui alt catalog decât catalogul curent, emiteți comanda

```
ls cale ← cale este calea relativă sau absolută a catalogului
```

Comanda *ls* acceptă o varietate de opțiuni care o determină să afișeze informații suplimentare. Indicatorul cel mai frecvent folosit este *-l*, care determină comanda *ls* să afișeze un singur fișier sau catalog pe fiecare linie de ieșire, astfel:

```
[mccartyb@athlon mccartyb]$ ls -l /home/mccartyb/public_html/
total 21
-rw-r--r-- 1 mccartyb  mccartyb    80 May 24 16:01 index.html
drwxrwxr-x 18 mccartyb  mccartyb   1024 Jun  2 09:54 php
-rwxr-xr-x 1 mccartyb  mccartyb   17915 May 24 15:55 php.tgz
```

Prima linie indică numărul de blocuri de catalog (i-noduri) asociate catalogului; în general, aceste informații nu sunt importante. Liniile următoare descriu un fișier sau un catalog. Dacă primul caracter al unei linii este o liniuță (-), atunci linia respectivă descrie un fișier; dacă primul caracter este litera d, linia respectivă descrie un catalog.

Următoarele două caractere indică permisiunile asociate fișierului sau catalogului. Primele trei caractere din acest grup specifică permisiunile posesorului; următoarele trei indică permisiunile membrilor grupului, iar ultimele trei indică permisiunile disponibile pentru alți utilizatori. Permisiunile sunt indicate folosindu-se următoarele coduri:

| Cod | Permisiune |
|-----|--------------------|
| r | Read (citire) |
| w | Write (scriere) |
| x | Execute (execuție) |

Apoi sunt date următoarele caracteristici, în ordinea apariției:

- Numărul legăturilor cu sistemul de fișiere asociate fișierului sau catalogului, parametru care, în general, nu este de interes
- Numele posesorului fișierului sau al catalogului
- Numele grupului posesor al fișierului sau al catalogului
- Dimensiunea fișierului sau a catalogului (în octeți)
- Data modificării fișierului sau a catalogului
- Numele fișierului sau al catalogului

Vizualizarea conținutului unui fișier și a datelor de ieșire ale unei comenzi

Pentru a vedea conținutul unui fișier text, emiteți comanda

```
more cale
```

unde *cale* este calea asociată fișierului. Comanda *more* vă permite să defilați prin fișier, pagină cu pagină. Apăsați pe tasta spațiu pentru a vă deplasa înainte. Pentru a încheia execuția programului, apăsați pe tasta q.

Dacă încercați să vizualizați conținutul unui fișier binar folosind comanda *more*, rezultatele vor fi lipsite de orice înțeles. În cel mai rău caz, conținutul fișierului poate afecta parametrii terminalului dumneavoastră, forțându-vă să închideți și apoi să redeschideți sesiunea de lucru pentru a vă continua activitatea.

Mai puteți folosi comanda *more* pentru a vizualiza datele de ieșire ale comenzii. Acest fapt este util când o comandă generează un volum apreciabil de date de ieșire. Pentru aceasta, emiteți comanda:

```
comanda | more
```

comanda simbolizează comanda care produce datele de ieșire, inclusiv opțiunile și argumentele conexe acesteia

Editarea unui fișier

UNIX acceptă o diversitate de editoare de text, precum vi și emacs. Totuși, începătorii se pot descurca folosind editorul *pico*, un „geamăn” al programului client de e-mail *pine*, care este deseori instalat de administratorii de sistem UNIX. Ca și *pine*, *pico* este controlat prin intermediul meniurilor și este simplu de utilizat. Pentru a lansa programul *pico*, emiteți comanda

```
pico
```

Pentru a edita un fișier existent, lansați *pico* prin emiterea comenzii

```
pico fișier ← fișier este calea spre fișierul care urmează a fi editat
```

Editorul *pico* include o documentație incorporată, concepută pentru a vă ajuta să învățați utilizarea programului respectiv. Folosiți tastele cu săgeți pentru a naviga până la un articol de meniu, după care selectați articolul de meniu prin apăsarea pe tasta ENTER.

Crearea unui catalog

Pentru a crea un catalog, emiteți comanda

```
mkdir cat ← cat este calea asociată catalogului
```

Ștergerea unui fișier

Pentru a șterge un fișier, emiteți comanda

```
rm fișier ← fișier este calea asociată fișierului
```

Avertisment

Deoarece UNIX nu dispune de un program de tip „recipient de deșuri” (Recycle Bin) de genul celui pus la dispoziție de Microsoft Windows, conținutul fișierelor șterse nu poate fi, în general, recuperat. Asigurați-vă că ștergeți numai fișiere de care nu mai aveți nevoie.

Ștergerea unui catalog

În cazul în care un catalog este vid – adică nu conține fișiere sau cataloage – îl puteți șterge prin emiterea comenzii

```
rmdir cat ← cat este calea asociată catalogului
```

În cazul în care catalogul nu este vid, puteți șterge catalogul și conținutul acestuia prin emiterea comenzii

```
rm -rf cat
```

unde *cat* este calea asociată catalogului.

Copierea unui fișier sau a unui catalog

Pentru a copia un fișier, emiteți comanda

```
cp fișier copie
```

unde *fișier* este calea asociată fișierului, iar *copie* este calea asociată copiei.

Pentru a copia un catalog și conținutul acestuia, emiteți comanda

```
cp -av cat copie
```

unde *cat* este calea asociată catalogului, iar *copie* este calea asociată copiei.

Modificarea numelui unui fișier sau al unui catalog

Pentru a modifica numele unui fișier sau al unui catalog, emiteți comanda

```
mv vechi nou
```

unde *vechi* este calea originală a fișierului sau a catalogului, iar *nou* este calea nouă.

Metacaracterele interfeței shell și globalizarea numelor

Prin încorporarea caracterelor speciale – numite *metacaractere* – în comenzi, puteți determina comenzile să opereze cu seturi întregi de fișiere. Această caracteristică UNIX, numită *globalizarea numelor*, este similară caracterelor de înlocuire folosite în MS-DOS, dar este considerabil mai complexă.

Metacaracterul *?* simbolizează un singur caracter, iar metacaracterul *** simbolizează un număr arbitrar de caractere (zero sau mai multe). UNIX definește metacaractere suplimentare, care nu sunt relevante pentru necesitățile unui începător.

Comanda

```
rm a?c
```

șterge fișierele din catalogul curent al căror nume are trei litere, începe cu litera *a* și se încheie cu litera *c*. De exemplu, dacă în catalogul curent există oricare dintre fișierele *aac*, *abc*, *acc* sau *adc*, comanda respectivă va șterge fișierul sau fișierele respective. Comanda

```
rm /home/dezordine/a*
```

șterge toate fișierele din catalogul */home/dezordine* ale căror nume încep cu litera *a*.

Comanda

```
rm -rf *
```

șterge toate fișierele și cataloagele din catalogul curent; această comandă și alte comenzi similare se vor folosi cu maximă atenție.

Puteți evita globalizarea numelor prin includerea unui argument între ghilimele gemene, fie ele simple sau duble. De exemplu, comanda

```
rm "a?c"
```

șterge fișierul al cărui nume are trei caractere, și anume caracterele *a*, *?* și *c*. Desigur, nu se recomandă utilizarea unor nume de fișiere de acest tip, mai ales dacă lucrați cu mai multe sisteme de operare.

Consultarea documentației UNIX pe suport electronic

UNIX include un sistem simplu de asistență on-line, care descrie modul de operare, opțiunile și argumentele comenzilor UNIX. De exemplu, pentru a învăța mai multe despre comanda *rm*, emiteți comanda

```
man rm
```

Informațiile furnizate de comanda *man* se numesc *pagină de manual*. Pentru a învăța mai multe despre comanda *man*, emiteți comanda:

```
man man
```

Paginile de manual sunt ordonate în secțiuni numerotate. Secțiunea 1 este dedicată comenzilor de genul celor folosite de utilizatorii obișnuiți. Informațiile despre un anumit subiect pot fi dispersate în mai multe secțiuni. Pentru a vizualiza pagina de manual din cadrul unei anumite secțiuni, emiteți comanda

```
man n subiect
```

unde *n* este numărul secțiunii, iar *subiect* este numele comenzii sau subiectul care vă interesează.

Raportarea gradului de utilizare a spațiului de pe disc

Pentru a vizualiza gradul de utilizare a spațiului din volumele discurilor existente în sistem, emiteți comanda

```
df -m
```

Datele de ieșire ale comenzii, care sunt asemănătoare cu lista prezentată mai jos, prezintă spațiul de pe disc utilizat și cel disponibil pentru fiecare sistem de fișiere, exprimat în MB:

| Filesystem | 1M-blocks | Used | Available | Use% | Mounted on |
|------------|-----------|------|-----------|------|------------|
| /dev/hda9 | 972 | 497 | 424 | 54% | / |
| /dev/hda1 | 996 | 529 | 467 | 54% | /dos |
| /dev/hda5 | 38 | 5 | 31 | 14% | /boot |
| /dev/hda6 | 1935 | 1456 | 399 | 79% | /usr |
| /dev/hda7 | 3874 | 3537 | 176 | 96% | /home |
| /dev/hda8 | 486 | 58 | 403 | 13% | /var |
| /dev/hda11 | 10066 | 7558 | 1985 | 80% | /space |

Pentru a vizualiza cantitatea de spațiu pe disc ocupată de anumite cataloage și fișiere, emiteți comanda

```
du -m cale
```

unde *cale* este numele fișierului sau al catalogului care vă interesează. Dacă specificați un catalog, comanda va afișa cantitatea de spațiu pe disc folosită de fiecare sub-catalog al catalogului.

Stabilirea posesorului unui fișier

La unele sisteme UNIX, posesorul unui fișier sau al unui catalog are permisiunea de a transfera dreptul de proprietate al fișierului sau al catalogului către un alt utilizator. Pentru aceasta, emiteți comanda

```
chown posesor cale
```

unde *cale* precizează calea asociată fișierului sau catalogului, iar *posesor* specifică numele de utilizator sau identificatorul noului posesor.

Pentru a specifica grupul posesor al unui fișier sau al unui catalog al cărui posesor sunteți, emiteți comanda

```
chgrp grup cale
```

unde *cale* precizează calea asociată fișierului sau catalogului, iar *grup* indică numele sau identificatorul numeric al noului grup posesor.

Configurarea permisiunilor pentru fișiere

Pentru a modifica permisiunile asociate unui fișier sau unui catalog pe care îl dețineți, emiteți comanda

```
chmod xxx cale
```

unde *cale* simbolizează calea asociată fișierului sau catalogului, iar *xxx* constă din trei cifre în octal care specifică permisiunile dorite. Prima cifră indică permisiunile asociate posesorului, cea de-a doua indică permisiunile asociate grupului din care face parte posesorul, iar a treia indică permisiunile asociate celorlalți utilizatori.

Fiecare cifră ia una din următoarele valori, care este echivalentă cu permisiunile indicate:

| Cifră | Permisiuni |
|-------|------------|
| 0 | --- |
| 1 | --x |
| 2 | -w- |
| 3 | -wx |
| 4 | r-- |
| 5 | r-x |
| 6 | rw- |
| 7 | rwX |

De exemplu, comanda

```
chmod 640 fisierulmeu
```

conferă posesorului acces de citire și scriere, membrilor grupului din care face parte posesorul – acces la citire, iar altor utilizatori – nici un fel de acces la fișierul numit *fișierulmeu*.

O a doua formă a comenzii *chmod* facilitează adăugarea, respectiv retragerea permisiunilor asociate unui fișier. Pentru detalii, consultați pagina de manual aferentă.

Găsirea unui fișier

Pentru a depista locația unui fișier în funcție de numele acestuia, emiteți comanda

```
find / -name gasestema
```

unde *gasestema* este numele fișierului. Căutarea începe de la catalogul rădăcină. În cazul în care doriți ca procesul de căutare să înceapă de la un alt catalog, specificați calea dorită în locul caracterului slash orientat înainte /. Când cunoașteți numai o parte a numelui fișierului, puteți folosi metacaractere pentru a specifica părțile pe care nu le cunoașteți; totuși, trebuie să includeți modelul rezultat între ghilimele, pentru a evita globalizarea numelor și pentru a prezenta modelul intact comenzii *find*. De exemplu, comanda următoare caută un fișier rezident undeva sub catalogul */home*, al cărui nume conține secvența *rosu*:

```
find /home -name "*rosu"
```

Unele sisteme UNIX acceptă comanda *locate*, care folosește o bază de date pentru a furniza rezultate mai rapid decât comanda *find*; totuși, rezultatele raportate nu sunt mai actuale decât baza de date asociată comenzii *locate*, care, în mod caracteristic, este actualizată zilnic. Pentru a găsi un fișier folosind comanda *locate*, emiteți comanda

```
locate gasestema
```

unde *gasestema* este numele fișierului.

Găsirea unui fișier care conține un text specificat

Pentru a găsi un fișier care conține un text specificat, emiteți comanda

```
grep -i text setfișiere
```

unde *text* reprezintă textul căutat, iar *setfișiere* constă din una sau mai multe căi care reprezintă fișiere. De exemplu, comanda

```
grep -i rosu fisier1 fisier2
```


caută textul roșu în fișierele `fișier1` și `fișier2`, text care poate apărea scris cu majuscule, minuscule sau combinat. O formă frecvent folosită a comenzii este

```
grep -i text *
```

care caută textul `text` în toate fișierele din catalogul curent.

În cazul în care `text` conține metacaractere sau constă din mai multe cuvinte, trebuie delimitat între ghilimele. Comanda `grep` pune la dispoziție numeroase opțiuni utile; pentru detalii, consultați pagina de manual aferentă comenzii.

Determinarea tipului unui fișier

Pentru a determina tipul unui fișier, emiteți comanda

```
file cale
```

unde `cale` este calea asociată fișierului. Comanda `file` nu se rezumă la a inspecta extensia numelui de fișier, dacă aceasta există, ci folosește euristici complicate pentru a determina tipul fișierului.

Compararea fișierelor text

Pentru a compara două fișiere text, emiteți comanda

```
diff --ignore-all-space cale1 cale2
```

unde `cale1` și `cale2` sunt căile asociate fișierelor text care urmează a fi comparate. Comanda raportează modificările necesare pentru a transforma un fișier în altul. Această comandă este deosebit de utilă pentru compararea versiunilor fișierelor HTML și a scripturilor PHP.

Lucrul cu fișiere comprimate

De obicei, fișierele UNIX sunt comprimate folosind una din următoarele două metode: metoda zip (comună în mediile Microsoft Windows) și metoda GNU zip. Fișierele zip pot conține fișiere și cataloage, în timp ce un fișier zip GNU poate conține un singur fișier.

Pentru a decompresa un fișier zip, emiteți comanda

```
unzip cale
```

unde `cale` este calea asociată fișierului comprimat.

Pentru a comprima un fișier sau un catalog folosind metoda zip, emiteți comanda

```
zip fișierzip setcale
```

unde `fișierzip` simbolizează calea asociată fișierului zip care urmează a fi creat, iar `setcale` reprezintă una sau mai multe fișiere sau cataloage care urmează a fi incluse

în fișierul zip. De exemplu, pentru a crea un fișier zip care conține fișierul `test` și conținutul catalogului `dosar`, puteți emite comanda

```
zip rezultat.zip test dosar
```

În mod convențional, numele fișierelor zip au ca extensie particula `.zip`.

Pentru a decompresa un fișier GNU zip, emiteți comanda

```
gunzip cale
```

unde `cale` este calea asociată fișierului comprimat.

Pentru a comprima un fișier folosind metoda GNU zip, emiteți comanda

```
gzip setcale
```

unde `setcale` specifică unul sau mai multe fișiere care urmează a fi comprimate. Pentru fiecare fișier specificat, programul de arhivare GNU zip creează un nou fișier, cu același nume ca originalul, dar având extensia `.gz`; fișierul original este șters. De exemplu, pentru a crea un fișier zip care conține fișierul `test`, puteți emite comanda

```
gzip test
```

Fișierul rezultat va avea numele `test.gz`.

Lucrul cu fișiere tar

Procedeele de combinare a mai multor fișiere și cataloage UNIX într-un singur fișier folosește mai frecvent comanda `tar` decât comanda `zip`. Fișierul rezultat în urma comenzii `tar` se numește *fișier tar* sau *tarball* (în traducere *bilă de gudron*). Pentru a crea un fișier tar, emiteți comanda

```
tar zcvf fișiertar.tgz setcale
```

unde `fișiertar.tgz` specifică numele fișierului tar care va fi creat, iar `setcale` specifică unul sau mai multe fișiere sau cataloage care vor fi incluse în fișierul tar. Unii utilizatori UNIX preferă să folosească extensia de fișier `.tar.gz` pentru un fișier tar. Opțiunea `z` impune utilizarea automată a programului GNU zip. Dacă se omite opțiunea `z` din comanda `zcvf`, va fi creat un fișier tar necomprimit, care are extensia `.tar`. În cazul în care se omite comprimarea în procesul de creare a unui fișier tar, puteți comprima ulterior fișierul folosind programul GNU zip.

Pentru a despacheta un fișier tar comprimat, emiteți comanda

```
tar zxvf fișiertar.tgz
```

unde `fișiertar.tgz` simbolizează numele fișierului tar. Pentru a despacheta un fișier tar decomprimit, omiteți opțiunea `z`, ca în instrucțiunea următoare:

```
tar xvf fișiertar.tar
```


Comanda `tar` pune la dispoziție numeroase alte opțiuni utile. Pentru mai multe detalii, consultați pagina de manual aferentă comenzii.

Transferul fișierelor spre și de la gazde aflate la distanță

Pentru a transfera fișierele *spre și de la* gazde aflate la distanță, puteți folosi comanda `ftp`, descrisă în Modulul 1. Totuși, un dezavantaj al utilizării acestei comenzi îl constituie faptul că FTP trimite numele de utilizator și parola dumneavoastră în rețea sub formă necriptată. Dacă atât gazda locală, cât și cea aflată la distanță acceptă SSH, puteți folosi programul `scp`, care criptează informațiile trimise prin rețea. Pentru a transfera un fișier spre un sistem aflat la distanță, prin intermediul SSH, emiteți comanda

```
scp fișier utilizator@gazda:cale
```

unde *fișier* este calea asociată fișierului care urmează a fi transferat, *utilizator* este numele contului de utilizator folosit pe gazda aflată la distanță, *gazda* este numele sau adresa IP a gazdei aflate la distanță, iar *cale* simbolizează calea de destinație pentru fișierul transferat.

Pentru a transfera un fișier de la un sistem aflat la distanță, prin intermediul SSH, emiteți comanda

```
scp utilizator@gazda:fișier cale
```

unde *utilizator* este numele contului de utilizator folosit pe gazda aflată la distanță, *gazda* este numele sau adresa IP a gazdei aflate la distanță, *fișier* este calea asociată fișierului care urmează a fi transferat, iar *cale* simbolizează calea de destinație pentru fișierul transferat.

Comanda `scp` pune la dispoziție numeroase alte opțiuni utile. Pentru detalii, consultați pagina de manual aferentă comenzii.

Anexa E: Caractere escape

| Secvența escape | Înlocuită prin |
|-------------------|--|
| <code>\n</code> | salt la linie nouă (ASCII 0x0A) |
| <code>\r</code> | retur de car (ASCII 0x0D) |
| <code>\t</code> | caracter de tabulare pe orizontală (ASCII 0x09) |
| <code>\\</code> | backslash |
| <code>\\$</code> | simbolul dolarului |
| <code>\"</code> | ghilimele duble |
| <code>\nnn</code> | Caracterul a cărui valoare ASCII este numărul <i>nnn</i> scris în baza 8 |
| <code>\xnn</code> | Caracterul a cărui valoare ASCII este numărul hexazecimal <i>nnn</i> |

Anexa F: ASCII

| Zecimal | Octal | Hexazecimal | Semnificație | Abreviere |
|---------|-------|-------------|----------------------|-----------|
| 0 | 0 | 0 | Null | NUL |
| 1 | 1 | 1 | Start of Heading | SOH |
| 2 | 2 | 2 | Start of text | STX |
| 3 | 3 | 3 | End of text | ETX |
| 4 | 4 | 4 | End of transmit | EOT |
| 5 | 5 | 5 | Enquiry | ENQ |
| 6 | 6 | 6 | Acknowledge | ACK |
| 7 | 7 | 7 | Audible Bell | BEL |
| 8 | 10 | 8 | Backspace | BS |
| 9 | 11 | 9 | Horizontal tab | HT |
| 10 | 12 | a | Line feed | LF |
| 11 | 13 | b | Vertical tab | VT |
| 12 | 14 | c | Form feed | FF |
| 13 | 15 | d | Carriage return | CR |
| 14 | 16 | e | Shift out | SO |
| 15 | 17 | f | Shift in | SI |
| 16 | 20 | 10 | Data link escape | DLE |
| 17 | 21 | 11 | Device control 1 | DC1 |
| 18 | 22 | 12 | Device control 2 | DC2 |
| 19 | 23 | 13 | Device control 3 | DC3 |
| 20 | 24 | 14 | Device control 4 | DC4 |
| 21 | 25 | 15 | Negative acknowledge | NAK |
| 22 | 26 | 16 | Synchronous idle | SYN |
| 23 | 27 | 17 | End transmit block | ETB |
| 24 | 30 | 18 | Cancel | CAN |
| 25 | 31 | 19 | End of medium | EM |
| 26 | 32 | 1a | Substitution | SUB |
| 27 | 33 | 1b | Escape | ESC |
| 28 | 34 | 1c | File separator | FS |
| 29 | 35 | 1d | Group separator | GS |
| 30 | 36 | 1e | Record separator | RS |
| 31 | 37 | 1f | Unit separator | US |
| 32 | 40 | 20 | Space | |
| 33 | 41 | 21 | ! | |
| 34 | 42 | 22 | " | |
| 35 | 43 | 23 | # | |
| 36 | 44 | 24 | \$ | |
| 37 | 45 | 25 | % | |
| 38 | 46 | 26 | & | |
| 39 | 47 | 27 | ' | |
| 40 | 50 | 28 | (| |
| 41 | 51 | 29 |) | |
| 42 | 52 | 2a | * | |

| Zecimal | Octal | Hexazecimal | Semnificație | Abreviere |
|---------|-------|-------------|--------------|-----------|
| 43 | 53 | 2b | + | |
| 44 | 54 | 2c | , | |
| 45 | 55 | 2d | - | |
| 46 | 56 | 2e | . | |
| 47 | 57 | 2f | / | |
| 48 | 60 | 30 | 0 | |
| 49 | 61 | 31 | 1 | |
| 50 | 62 | 32 | 2 | |
| 51 | 63 | 33 | 3 | |
| 52 | 64 | 34 | 4 | |
| 53 | 65 | 35 | 5 | |
| 54 | 66 | 36 | 6 | |
| 55 | 67 | 37 | 7 | |
| 56 | 70 | 38 | 8 | |
| 57 | 71 | 39 | 9 | |
| 58 | 72 | 3a | : | |
| 59 | 73 | 3b | ; | |
| 60 | 74 | 3c | < | |
| 61 | 75 | 3d | = | |
| 62 | 76 | 3e | > | |
| 63 | 77 | 3f | ? | |
| 64 | 100 | 40 | @ | |
| 65 | 101 | 41 | A | |
| 66 | 102 | 42 | B | |
| 67 | 103 | 43 | C | |
| 68 | 104 | 44 | D | |
| 69 | 105 | 45 | E | |
| 70 | 106 | 46 | F | |
| 71 | 107 | 47 | G | |
| 72 | 110 | 48 | H | |
| 73 | 111 | 49 | I | |
| 74 | 112 | 4a | J | |
| 75 | 113 | 4b | K | |
| 76 | 114 | 4c | L | |
| 77 | 115 | 4d | M | |
| 78 | 116 | 4e | N | |
| 79 | 117 | 4f | O | |
| 80 | 120 | 50 | P | |
| 81 | 121 | 51 | Q | |
| 82 | 122 | 52 | R | |
| 83 | 123 | 53 | S | |
| 84 | 124 | 54 | T | |
| 85 | 125 | 55 | U | |
| 86 | 126 | 56 | V | |
| 87 | 127 | 57 | W | |
| 88 | 130 | 58 | X | |
| 89 | 131 | 59 | Y | |

| Zecimal | Octal | Hexazecimal | Semnificație | Abreviere |
|---------|-------|-------------|--------------|-----------|
| 90 | 132 | 5a | Z | |
| 91 | 133 | 5b | [| |
| 92 | 134 | 5c | \ | |
| 93 | 135 | 5d |] | |
| 94 | 136 | 5e | ^ | |
| 95 | 137 | 5f | - | |
| 96 | 140 | 60 | , | |
| 97 | 141 | 61 | a | |
| 98 | 142 | 62 | b | |
| 99 | 143 | 63 | c | |
| 100 | 144 | 64 | d | |
| 101 | 145 | 65 | e | |
| 102 | 146 | 66 | f | |
| 103 | 147 | 67 | g | |
| 104 | 150 | 68 | h | |
| 105 | 151 | 69 | i | |
| 106 | 152 | 6a | j | |
| 107 | 153 | 6b | k | |
| 108 | 154 | 6c | l | |
| 109 | 155 | 6d | m | |
| 110 | 156 | 6e | n | |
| 111 | 157 | 6f | o | |
| 112 | 160 | 70 | p | |
| 113 | 161 | 71 | q | |
| 114 | 162 | 72 | r | |
| 115 | 163 | 73 | s | |
| 116 | 164 | 74 | t | |
| 117 | 165 | 75 | u | |
| 118 | 166 | 76 | v | |
| 119 | 167 | 77 | w | |
| 120 | 170 | 78 | x | |
| 121 | 171 | 79 | y | |
| 122 | 172 | 7a | z | |
| 123 | 173 | 7b | { | |
| 124 | 174 | 7c | | |
| 125 | 175 | 7d | } | |
| 126 | 176 | 7e | ~ | |
| 127 | 177 | 7f | Delete | |

Anexa G: Operatori PHP

| Precedență | Asociativitate | Operatori | Semnificație |
|------------|----------------|-----------|--------------------------------------|
| 20 | Nici una | new | Constructor de obiecte |
| 19 | La dreapta | [] | Indice |
| 18 | La dreapta | ! | NU logic |
| | | ~ | NU la nivel de bit |
| | | ++ | Incrementare |
| | | -- | Decrementare |
| | | (int) | Conversie forțată la întreg |
| | | (double) | Conversie forțată la dublu |
| | | (string) | Conversie forțată la șir |
| | | (array) | Conversie forțată la tablou |
| | | (object) | Conversie forțată la obiect |
| | | @ | Controlul erorii |
| 17 | La stânga | * | Înmulțire |
| | | / | Împărțire |
| | | % | Operatorul modulo |
| 16 | La stânga | + | Adunare |
| | | - | Scădere |
| | | . | Concatenare a șirurilor |
| 15 | La stânga | << | Deplasare la stânga la nivel de bit |
| | | >> | Deplasare la dreapta la nivel de bit |
| 14 | Nici una | < | Mai mic |
| | | <= | Mai mic sau egal |
| | | > | Mai mare |
| | | >= | Mai mare sau egal |
| 13 | Nici una | == | Egal |
| | | != | Diferit |
| | | === | Identic |
| | | !== | Nu este identic |
| 12 | La stânga | & | ȘI la nivel de bit |
| 11 | La stânga | ^ | SAU exclusiv la nivel de bit |
| 10 | La stânga | | SAU la nivel de bit |
| 9 | La stânga | && | ȘI logic |
| 8 | La stânga | | SAU logic |
| 7 | La stânga | ?: | Condițional |
| 6 | La stânga | = | Atribuire |
| | | += | Atribuire de adunare |
| | | -= | Atribuire de scădere |
| | | *= | Atribuire de înmulțire |
| | | /= | Atribuire de împărțire |

| Precedență | Asociativitate | Operatori | Semnificație |
|------------|----------------|-----------|--|
| | | . | Atribuire de concatenare |
| | | %= | Atribuire de modulo |
| | | &= | Atribuire ȘI la nivel de bit |
| | | = | Atribuire SAU la nivel de bit |
| | | ^= | Atribuire SAU exclusiv la nivel de bit |
| | | ~= | Atribuire NU la nivel de bit |
| | | <<= | Atribuire deplasare la stânga la nivel de bit |
| | | >>= | Atribuire deplasare la dreapta la nivel de bit |
| 5 | La dreapta | print | Afișare |
| 4 | La stânga | and | ȘI logic |
| 3 | La stânga | xor | SAU exclusiv logic |
| 2 | La stânga | sau | SAU logic |
| 1 | La stânga | , | Virgulă |

Anexa H: Securitate

Securitatea reprezintă un subiect complex, care nu este adecvat pentru un programator începător. Totuși, chiar și programatorii PHP începători doresc să creeze aplicații și să le pună în funcțiune. Un aspect nefericit al rețelelor de calculatoare este acela că aplicațiile disponibile prin intermediul unei rețele vor fi supuse atacurilor, atât din partea unor persoane rău-intenționate, cât și din partea curioșilor. Această anexă are menirea de a atrage atenția cititorului asupra unor aspecte și resurse legate de securitate care nu au fost prezentate în cartea de față. Cititorul atent la posibilitatea existenței breșelor în sistemul de securitate trebuie să fie motivat pentru a învăța mai multe despre acestea, precum și despre modul de a le preveni. Cititorii interesați în desfășurarea de aplicații Internet care prelucrează date importante – cum sunt tranzacțiile de afaceri – trebuie să-și continue studiul legat de PHP și de securitate sau să solicite consultanță din partea unui expert în probleme de securitate.

Pericole și riscuri

Un principiu fundamental al securității produselor software este acela că nici un sistem nu este absolut sigur. În consecință, securitatea nu este o chestiune de tipul „totul sau nimic”. În schimb, politicile și mecanismele de securitate trebuie să fie concepute în funcție de:

- Frecvența și natura amenințărilor la adresa securității
- Posibilele consecințe ale unei breșe de securitate

Altfel spus, securitatea este legată mai mult de descurajare decât de prevenire. Dacă dispune de timp și de experiență, un hacker motivat are toate șansele de a compromite orice sistem. Scopul securității este de a ridica ștacheta elementelor necesare pentru a compromite un sistem și de a impune unui potențial hacker să piardă o cantitate de timp mai mare decât cea rezonabilă din punctul de vedere al recompensei pe care un hacker speră să o obțină în urma unei pătrunderi reușite în sistem.

Hackerii pot fi motivați de perspectiva unor câștiguri financiare, de dorința de răzbunare, de o simplă curiozitate sau de o varietate de alte motive. Unii hackeri dispun de o experiență considerabilă în materie de programe. Alții, cunoscuți și sub numele de *script kiddies* (începători), de abia au învățat să folosească instrumentele și procedurile create de alții. Cu toate acestea, un hacker hotărât și cu un oarecare grad de cunoștințe poate fi un adversar extrem de redutabil.

O altă modalitate de a caracteriza un pericol la adresa securității este din punctul de vedere al modalității în care pericolul afectează un sistem țintă, și anume *tipul*

pericolului. O aplicație software în rețea este susceptibilă la o diversitate de tipuri de pericole. Printre acestea se numără:

- Dezvăluirea unor date confidențiale
- Modificarea sau distrugerea datelor
- Refuzul de a executa un serviciu
- Repudierea tranzacțiilor

Dezvăluirea datelor confidențiale este un pericol deosebit de frecvent întâlnit. Printre variantele cele mai caracteristice ale acestuia se numără publicarea mesajelor de e-mail private și a informațiilor privind cartea de credit. Protecția împotriva acestui pericol implică o administrare de sistem și o programare conform normelor în vigoare.

Modificarea sau distrugerea datelor reprezintă un alt pericol comun. Deseori, acesta ia forma deteriorării unui sit Web. Virușii de calculator care modifică datele reprezintă o altă formă comună de atac. Din nou, protecția împotriva acestor pericole implică respectarea normelor recunoscute privind programarea și administrarea sistemelor.

Refuzul de a executa un serviciu este un alt tip de pericol care a devenit extrem de frecvent întâlnit. Dacă dezvăluirea informațiilor confidențiale și modificarea sau distrugerea datelor sunt deseori rezultatul unei breșe de securitate înlesnite printr-o acțiune sau o omisiune a unui programator sau a unui administrator de sistem, atacurile prin refuzul de a executa un serviciu* nu implică asemenea puncte slabe. În prezent, majoritatea sistemelor sunt susceptibile la atacuri prin refuzul de a executa un serviciu, care bombardează o gazdă cu cereri ilicite de serviciu într-o asemenea măsură, încât gazda devine incapabilă să răspundă la cererile efective. În prezent, se lucrează la identificarea unor modalități de contracarare a acestor atacuri.

Repudierea tranzacțiilor este un pericol la adresa securității asociat cu noțiunea de comerț electronic și constă în aceea că una din părțile implicate într-o tranzacție nu recunoaște, la un moment ulterior, că a autorizat tranzacția. Protecția împotriva acestui pericol implică o proiectare a aplicațiilor care să includă tehnici criptografice, precum și de altă natură, pentru autentificarea identității părților implicate într-o tranzacție.

Este important de reținut că severitatea consecințelor care pot rezulta dintr-o breșă de securitate poate varia într-o gamă foarte largă. De exemplu, consecințele care pot decurge din dezvăluirea de informații confidențiale pot varia de la ușor jenante – în cazul publicării unui mesaj de e-mail privat care conține și unele expresii mai „tari” – la catastrofale – prin publicarea unor planuri de afaceri confidențiale sau a unor documente care constituie proprietate intelectuală.

* În original *denial of service* - abreviat DOS. Nu se va confunda cu MS-DOS, adică Microsoft Disk Operating System - N.T.

Contramăsuri

Această secțiune explică unele dintre contramăsurile frecvent întrebuințate pentru a reduce riscul apariției unei breșe de securitate. Tratamentul propus nu este nici complet și nici foarte amănunțit, fiind destinat a-l pune la curent pe cititor cu privire la unele posibilități mai evidente, nu de a furniza un îndrumar pentru asigurarea securității aplicațiilor. Secțiunea următoare conține trimiteri spre numeroase surse mai bine documentate.

Autentificarea și autorizarea utilizatorilor

Când o aplicație conține funcții care nu sunt destinate utilizării de către publicul larg, aplicația trebuie să includă mecanisme pentru identificarea, autentificarea și autorizarea utilizatorilor sistemului. Unii începători scriu aplicații care solicită utilizatorilor să se identifice. Deoarece un utilizator rău intenționat va prezenta o identitate care nu-i aparține, simpla identificare nu este suficientă.

Autentificarea implică verificarea identității unui utilizator. Cea mai simplă posibilitate de autentificare constă în utilizarea unei parole private. Totuși, alte mijloace – cum este procedeul *întrebare și răspuns*, în care utilizatorul primește o întrebare dintr-un set predeterminat de întrebări – sunt de asemenea posibile.

Autorizarea asociază privilegiile cu identitatea unui utilizator. De exemplu, unii utilizatori pot avea permisiunea de a vizualiza datele privind salariile într-o aplicație care manipulează datele referitoare la angajați, dar numai utilizatorii cu privilegii speciale pot modifica datele referitoare la salarii.

Stocarea datelor privind autorizarea și autentificarea într-o bază de date permite utilizatorilor desemnați să întrețină aceste informații și implicit să actualizeze în permanență datele. Breșele de securitate pot surveni atunci când nu sunt retrase autorizațiile utilizatorilor care nu mai sunt asociați unei anumite companii sau unui anumit proiect. Informațiile privind parolele trebuie să fie stocate în formă criptată, astfel încât nici măcar administratorii de sistem să nu poată falsifica identitățile.

Suspectarea datelor introduse de utilizatori

Așa cum s-a explicat în Modulul 14, datele provenite de la utilizatori trebuie privite cu neîncredere. Furnizarea de date care conțin metacaractere reprezintă un mijloc frecvent folosit de compromitere a securității aplicației. Când scrieți aplicații PHP bazate pe rețea, trebuie să folosiți funcția `addslashes()` și alte mijloace pentru a vă asigura că datele introduse de utilizator nu inițiază operații de prelucrare neprevăzute.

Întrebuințarea unor măsuri criptografice

Această carte nu se referă la suportul PHP pentru funcții criptografice. Totuși, utilizarea unor asemenea funcții este importantă pentru protejarea securității.

O formă populară de securitate este așa-numitul server Web sigur, care schimbă date cu browserele Web prin intermediul HTTPS, o variantă criptată a protocolului HTTP comun. Tranzacțiile folosite în comerțul electronic se derulează deseori utilizând protocolul HTTPS.

Printre alte tehnologii criptografice disponibile se numără criptografia cu cheie publică și cu cheie privată, care se poate folosi pentru autentificarea utilizatorilor și a mesajelor, precum și pentru schimbul de informații în condiții de siguranță. Între tehnologiile care se pot dovedi utile se numără următoarele:

- Algoritmi MD5
- Criptare RSA
- Criptare PG pentru PGP și GNU

Configurarea adecvată a serverelor

Datorită modului în care PHP interacționează cu un server Web, configurarea inadecvată a serverului Web poate avea consecințe grave asupra securității PHP. Dacă sunteți administratorul de sistem al unei gazde care furnizează servicii de Web, trebuie să fiți la curent cu opțiunile serverului Web relative la securitate. De asemenea, trebuie să vă procurați și să instalați remedii pentru sistemul de operare, serverul Web și PHP care au implicații pentru securitatea sistemului și a rețelei.

Un aspect important al securității serverului Web constă în prevenirea accesului fără autorizație la fișierele sursă și la fișierele de date. Trebuie să configurați serverul Web astfel încât să permită browserelor Web să obțină acces numai la cataloagele selectate, precum și să vă asigurați că fișierele sursă și fișierele de date se găsesc în alte cataloage decât acestea.

Asigurarea bazelor de date

Numele de utilizator și parolele folosite pentru conectarea la bazele de date sunt, de obicei, codate hard în scripturile PHP. Dacă un hacker poate determina numele de utilizator și parola, atunci se poate conecta la baza de date și poate executa orice operație neautorizată dorește. De exemplu, un hacker poate șterge toate rândurile unui sau mai multor tabele din baza de date.

O modalitate de a proteja numele de utilizator și parolele constă în a le insera în fișiere care sunt accesibile prin intermediul instrucțiunilor include sau require. Nu este necesar — și nici *nu trebuie* — ca aceste fișiere să se găsească în cataloage accesibile din Web. Prin amplasarea acestor informații într-un catalog

care nu este accesibil din Web, devine mult mai dificil pentru un hacker să le descopere conținutul.

Resurse

Pentru o introducere în tematica securității pentru rețele, abordată din punctul de vedere al începătorilor, consultați volumul *Network Security: A Beginner's Guide*, de Eric Maiwald (Osborne McGraw-Hill, 2001). Pentru a învăța mai multe despre asigurarea sistemelor UNIX împotriva pericolelor locale și de la distanță, consultați cartea *UNIX System Security Tools*, de Seth Ross (Osborne McGraw-Hill, 1999). Informații referitoare la modul de configurare a serverului Apache se găsesc în volumul *Administering Apache*, de Mark Arnold (Osborne McGraw-Hill, 2000).

Anexa I:

Funcții PHP

Această anexă descrie principalele funcții PHP, inclusiv pe cele folosite în carte. PHP include numeroase alte funcții, care nu pot fi descrise în anexa de față, datorită spațiului limitat. În anexă puteți găsi rapid o funcție utilă sau puteți găsi ordinea și tipurile argumentelor unei funcții. Pentru informații suplimentare referitoare la funcțiile descrise în anexă, precum și la funcțiile care nu sunt prezentate aici, consultați manualul PHP pe suport electronic din situl Web PHP, la adresa <http://www.php.net>.

Funcțiile sunt descrise prin intermediul unui prototip de funcție, adică aceeași metodă folosită în manualul PHP pe suport electronic. Prototipurile funcțiilor sunt oarecum mai complexe decât modelele sintactice folosite în restul cărții, dar prototipurile funcțiilor furnizează mai multe informații decât șabloanele sintactice simple. Iată un exemplu de prototip de funcție:

```
int area (double lungime [, double latime])
```

Acest prototip descrie o funcție numită *area*, care returnează o valoare de tipul *int*. Primul argument al funcției se numește *lungime* și este de tip *double*. Desigur, puteți invoca funcția folosind un argument având orice nume. Numele *lungime* este un nume inventat – un parametru formal – folosit pentru a se face referire la argument. Numele argumentului apare scris folosind caractere cursive, pentru a se arăta că poate fi substituit.

Parantezele drepte arată că al doilea argument – un argument de tip *double* numit *latime* – este opțional. Dacă argumentul este specificat, trebuie să fie inserată și virgula care îl precede. Când un argument poate apărea de mai multe ori, este inserat simbolul *...*. De exemplu, prototipul funcției *array()* se prezintă astfel:

```
array array ([mixed...])
```

Prototipul arată că funcția *array()* returnează o valoare de tipul *array* și că funcția acceptă zero sau mai multe argumente de orice tip.

Când un argument poate avea orice tip, specificația argumentului conține cuvântul cheie *mixed* (combinat). Când tipul unui argument poate fi unul din mai multe tipuri specificate, tipurile sunt date și separate printr-o bară verticală. De exemplu, un argument de tip *int|double* poate avea tipul *int* sau tipul *double*. Tipul special *resource* este folosit pentru a se face referire la valorile returnate de funcții precum *mysql_connect()*.

Funcții tablou

Prototipul funcției

```
array array ([mixed...])
array array_count_values (array intrare)
array array_diff (array tablou1, array
tablou2 [, array ...])
array array_filter (array intrare
[, mixed apel_invers])
array array_flip (array trans)
array array_intersect (array tablou1,
array tablou2 [, array ...])
array array_keys (array intrare
[, mixed valoare_cautare])
array array_merge (array tablou1,
array tablou2 [, array ...])
array array_merge_recursive (array
tablou1, array tablou2 [, array ...])
bool array_multisort (array tb1
[, mixed argument [, mixed ...
[, array ...]])
array array_pad (array intrare, int
dim_completare, mixed val_completare)
mixed array_pop (array tablou)
int array_push (array tablou, mixed var
[, mixed ...])
mixed array_rand (array intrare
[, int num_req])
mixed array_reduce (array intrare mixed
apel_invers [, int initial])
array array_reverse (array tablou
[, bool pastrare_chei])
mixed array_search (mixed ac, array
car_cu_fan, bool strict)

mixed array_shift (array tablou)

array array_slice (array tablou,
int offset [, int lungime])
array array_splice (array intrare,
int offset [, int lungime
[, array inlocuire]])
mixed array_sum (array tablou)
array array_unique (array tablou)
```

Descriere

Returnează un tablou al parametrilor.

Numără valorile dintr-un tablou.

Calculează diferența între două sau mai multe tablouri.

Filtrează elementele unui tablou folosind o funcție cu apel invers.

Inversează ordinea valorilor dintr-un tablou.

Determină elementele pe care le au în comun două sau mai multe tablouri.

Returnează cheile unui tablou.

Fuzionează două sau mai multe tablouri.

Fuzionează în mod recursiv două sau mai multe tablouri.

Sortează unul sau mai multe tablouri de una sau mai multe dimensiuni.

Completează tabloul până la lungimea specificată.

Extrage elementul de la sfârșitul tabloului.

Introduce unul sau mai multe elemente la sfârșitul tabloului.

Selectează una sau mai multe intrări aleatoare dintr-un tablou.

Aplică o funcție fiecărui element al unui tablou și returnează rezultatul.

Returnează un tablou cu elementele în ordine inversată.

Caută o valoare dată într-un tablou și returnează cheia corespunzătoare, dacă găsește valoarea.

Extrage un element de la începutul unui tablou.

Extrage o „felie” dintr-un tablou.

Elimină și înlocuiește o porțiune dintr-un tablou.

Calculează suma valorilor dintr-un tablou.

Elimină valorile care se repetă dintr-un tablou.

Prototipul funcției

```

int array_unshift (array tablou,
mixed var [, mixed ...])
array array_values (array intrare)
int array_walk (array tablou, string
functie, mixed date_utilizator)
void arsort (array tablou [, int
sortare_indicatori])
void asort (array tablou [, int
sortare_indicatori])
array compact (mixed numevar
[, mixed ...])
int count (mixed var)
mixed current (array tablou)
array each (array tablou)

mixed end (array tablou)

int extract (array var_tablou [, int
tip_extract [, string prefix]])
bool in_array (mixed ac, array
car_cu_fan [, bool strict])
mixed key (array tablou)
int krsort (array tablou [, int
indicatoare_sortare])
int ksort (array tablou [, int
indicatoare_sortare])
void list (...)
void natcasesort (array tablou)

void natsort (array tablou)

mixed next (array tablou)
mixed pos (array tablou)
mixed prev (array tablou)

array range (int mic, int mare)

mixed reset (array tablou)

void rsort (array tablou [, int
indicatoare_sortare])

```

Descriere

Inserează unul sau mai multe elemente la începutul unui tablou.

Returnează toate valorile dintr-un tablou.

Aplică o funcție fiecărui membru al unui tablou.

Sortează un tablou în ordine inversă, menținând asocierea indexurilor.

Sortează un tablou, menținând asocierea indexurilor.

Creează un tablou care conține valorile variabilelor specificate.

Numără elementele unei variabile.

Returnează elementul curent al unui tablou.

Returnează următoarea pereche cheie-valoare dintr-un tablou.

Stabilește poziția pointerului intern al tabloului la ultimul element din tablou.

Importă variabile, specificate de un tablou, în tabelul cu simboluri.

Returnează TRUE dacă o valoare specificată există în cadrul unui tablou.

Preia o cheie dintr-un tablou asociativ.

Sortează un tablou în ordine inversă, în funcție de cheie.

Sortează un tablou în funcție de cheie.

Atribue valori unei serii de variabile.

Sortează un tablou folosind o ordine naturală insensibilă la diferența dintre majuscule și minuscule.

Sortează un tablou folosind o ordine naturală.

Avansează pointerul intern al unui tablou.

Preia elementul curent al unui tablou.

Deplasează înapoi pointerul intern al tabloului.

Creează un tablou care conține un domeniu de valori întregi.

Poziționează pointerul intern al tabloului în dreptul primului element al acestuia.

Sortează un tablou în ordine inversă.

Prototipul funcției

```

void shuffle (array tablou)

int sizeof (array tablou)

void sort (array tablou [, int
indicatoare_sortare])
void uasort (array tablou, function
functie_comp)
void uksort (array tablou, function
functie_comp)

void usort (array tablou, function
functie_comp)

```

Descriere

Distribuie elementele dintr-un tablou într-o ordine aleatorie.

Returnează numărul elementelor dintr-un tablou.

Sortează un tablou.

Sortează un tablou folosind o funcție de comparație definită de utilizator.

Sortează un tablou în funcție de chei, folosind o funcție de comparație definită de utilizator.

Sortează un tablou în funcție de valori, folosind o funcție de comparație definită de utilizator.

Funcții de verificare a ortografiei**Prototip de funcție**

```

int aspell_new (string master,
string personal)
bool aspell_check (int legatura_dictionar, Verifică ortografia unui cuvânt.
string cuvnt)
bool aspell_check_raw (int
legatura_dictionar, string cuvnt)
bool aspell_suggest (int
legatura_dictionar, string cuvnt)

```

Descriere

Încarcă un dicționar ortografic.

Verifică ortografia unui cuvânt.

Verifică ortografia unui cuvânt, fără a-i modifica dimensiunea literelor și fără a-l amputa.

Sugerează posibile variante ortografice ale unui cuvânt.

Funcții calendar**Prototip de funcție**

```

int easter_date (int an)

int easter_days (int an)

int frenchtojd (int luna, int zi, int an)

int gregoriantojd (int luna, int zi,
int an)
mixed jddayofweek (int zi_iulian, int mod)
string jdmonthname (int zi_iulian,
int mod)

```

Descriere

Obține o amprentă de timp UNIX pentru miezul nopții de Paști a unui an dat.

Preia intervalul de timp, exprimat în zile, cuprins între 21 martie și ziua de Paști a unui an dat.

Convertește o dată din calendarul republican francez într-un număr de zile din calendarul iulian.

Convertește o dată din calendarul gregorian într-un număr de zile din calendarul iulian.

Returnează ziua din săptămână.

Returnează numele unei luni.

Prototip de funcție

| | |
|--|--|
| <code>string jdtofrench (int numar_zile_iulian)</code> | Convertește un număr de zile din calendarul iulian în calendarul republican francez. |
| <code>string jdtogregorian (int zi_iulian)</code> | Convertește un număr de zile din calendarul iulian într-o dată din calendarul gregorian. |
| <code>string jdtojewish (int zi_iulian)</code> | Convertește un număr de zile din calendarul iulian în calendarul evreiesc. |
| <code>string jdtojulian (int zi_iulian)</code> | Convertește un număr de zile din calendarul iulian într-o dată din calendarul iulian. |
| <code>int jdtounix (int zi_iulian)</code> | Convertește un număr de zile din calendarul iulian într-o amprentă de timp UNIX. |
| <code>int jewishtojd (int luna, int zi, int an)</code> | Convertește o dată din calendarul evreiesc într-un număr de zile din calendarul iulian. |
| <code>int juliantojd (int luna, int zi, int an)</code> | Convertește o dată din calendarul iulian într-un număr de zile din calendarul iulian. |
| <code>int unixtojd ([int amprenta_timp])</code> | Convertește o amprentă de timp UNIX într-un număr de zile din calendarul iulian. |

Funcții cu clase și obiecte

Prototipul funcției

| | |
|---|--|
| <code>mixed call_user_method_array (string metoda, object obiect [, array parametri])</code> | Apelează o metodă utilizator cu un tablou de parametri specificat. |
| <code>mixed call_user_method (string metoda, object obiect[, mixed param [, mixed ...]])</code> | Apelează o metodă utilizator asupra unui obiect specificat. |
| <code>bool class_exists (string nume_clasa)</code> | Verifică dacă a fost definită o clasă. |
| <code>string get_class (object obiect)</code> | Returnează numele clasei unui obiect. |
| <code>array get_class_methods (string nume_clasa)</code> | Returnează un tablou care conține numele metodelor unei clase. |
| <code>array get_class_vars (string nume_clasa)</code> | Returnează un tablou care conține proprietățile prestabilite ale unei clase. |
| <code>array get_declared_classes ()</code> | Returnează un tablou care conține numele claselor definite. |
| <code>array get_object_vars (object obiect)</code> | Returnează un tablou asociativ al proprietăților obiectului. |
| <code>string get_parent_class (object obiect)</code> | Returnează numele clasei părinte a obiectului. |
| <code>bool method_exists (object obiect, string metoda)</code> | Verifică dacă metoda unei clase există. |

Funcții de tip caracter

Prototipul funcției

| | |
|--|---|
| <code>bool ctype_alnum (string c)</code> | Descriere
Caută unul sau mai multe caractere alfanumerice. |
|--|---|

Descriere**Descriere****Descriere****Prototipul funcției**

| | |
|---|---|
| <code>bool ctype_alpha (string c)</code> | Descriere
Caută unul sau mai multe caractere alfabetice. |
| <code>bool ctype_cntrl (string c)</code> | Caută unul sau mai multe caractere de control. |
| <code>bool ctype_digit (string c)</code> | Caută unul sau mai multe caractere numerice. |
| <code>bool ctype_lower (string c)</code> | Caută unul sau mai multe caractere scrise cu minuscule. |
| <code>bool ctype_graph (string c)</code> | Caută unul sau mai multe caractere care pot fi afișate, cu excepția caracterului spațiu. |
| <code>bool ctype_print (string c)</code> | Caută unul sau mai multe caractere care pot fi afișate. |
| <code>bool ctype_punct (string c)</code> | Caută unul sau mai multe caractere care pot fi afișate, caractere care nu sunt nici spații albe, nici caractere alfanumerice. |
| <code>bool ctype_space (string c)</code> | Caută unul sau mai multe caractere de tip spații albe. |
| <code>bool ctype_upper (string c)</code> | Caută unul sau mai multe caractere scrise cu majuscule. |
| <code>bool ctype_xdigit (string c)</code> | Caută unul sau mai multe caractere care reprezintă o cifră hexazecimală. |



Funcții de tip dată și oră

Prototipul funcției

| | |
|--|--|
| <code>int checkdate (int luna, int zi, int an)</code> | Descriere
Validează o dată din calendarul gregorian. |
| <code>string date (string format [, int amprenta_timp])</code> | Formatează o dată și o oră locală. |
| <code>array getdate ([int amprenta_timp])</code> | Obține informații despre dată și oră. |
| <code>array gettimeofday ()</code> | Obține ora curentă. |
| <code>string gmtime (string format [, int amprenta_timp])</code> | Formatează o dată/oră GMT. |
| <code>int gmmktime (int ora, int minut, int secunda, int luna, int zi, int an [, int is_dst])</code> | Obține o amprentă de timp UNIX pentru o dată GMT. |
| <code>string gmstrftime (string format [, int amprenta_timp])</code> | Formatează o dată/oră GMT în conformitate cu parametrii locali. |
| <code>array localtime ([int amprenta_timp [, bool este_asociativ]])</code> | Obține ora locală. |
| <code>string microtime ()</code> | Returnează amprenta de timp UNIX curentă, în microsecunde. |
| <code>int mktime (int ora, int minut, int secunda, int zi, int an, [, int is_dst])</code> | Obține o amprentă de timp UNIX pentru dată. |
| <code>string strftime (string format [, int amprenta_timp])</code> | Formatează o dată/oră locală în conformitate cu parametrii locali. |

Prototipul funcției

```
int strtotime (string ora [, int acum])
```

```
int time ()
```

Descriere

Convertește o dată/oră în format textual britanic într-o amprentă de timp UNIX.
Returnează o amprentă de timp curentă UNIX.

Funcții de manipulare a cataloagelor

Prototipul funcției

```
bool chroot (string cat)
```

```
bool chdir (string cat)
```

```
newdir (string cat)
```

```
void closedir (resource cat_manip)
```

```
string getcwd ()
```

```
resource opendir (string cale)
```

```
string readdir (resource cat_manip)
```

```
void rewinddir (resource cat_manip)
```

Descriere

Înlocuiește catalogul rădăcină al procesului curent.
Înlocuiește catalogul curent de lucru.
Creează un obiect catalog.
Închide o variabilă de manipulare a cataloagelor.
Obține catalogul curent de lucru.
Deschide o variabilă de manipulare a cataloagelor.
Citește o intrare din variabila de manipulare a cataloagelor.
Deplasează înapoi o variabilă de manipulare a cataloagelor.

Funcții de tratare și de consemnare a erorilor

Prototipul funcției

```
int error_log (string mesaj, int tip_mesaj [, string destinatie [, string extra_antete]])
```

```
int error_reporting ([int nivel])
```

```
void restore_error_handler ()
```

```
string set_error_handler (string functie_tratare_eroare)
```

```
void trigger_error (string mesaj_eroare [, int tip_eroare])
```

```
void user_error (string mesaj_eroare [, int tip_eroare])
```

Descriere

Trimite un mesaj de eroare într-un fișier jurnal.
Specifică erorile PHP care sunt raportate.
Restaurează funcția anterioară de tratare a erorilor.
Configurează o funcție de tratare a erorilor definită de utilizator.
Generează un mesaj de eroare la nivel de utilizator.
Generează un mesaj de eroare la nivel de utilizator.

Funcții ale sistemului de fișiere

Prototipul funcției

```
string basename (string cale)
```

```
int chgrp (string nume_fisier, mixed grup)
```

```
int chmod (string nume_fisier, int mod)
```

Descriere

Returnează componenta nume de fișier a căii.
Modifică grupul proprietar al fișierului.
Modifică permisiunile unui fișier.

Prototipul funcției

```
int chown (string nume_fisier, mixed utilizator)
```

```
void clearstatcache ()
```

```
int copy (string sursa, string destinatar)
```

```
string dirname (string cale)
```

```
float diskfreespace (string catalog)
```

```
bool fclose (int fp)
```

```
int feof (int fp)
```

```
int fflush (int fp)
```

```
string fgetc (int fp)
```

```
array fgetcsv (int fp, int lungime [, string delimitator])
```

```
string fgets (int fp, int lungime)
```

```
string fgetss (int fp, int lungime [, string etichete_permise])
```

```
array file (string nume_fisier [, int foloseste_cale_include])
```

```
bool file_exists (string nume_fisier)
```

```
int fileatime (string nume_fisier)
```

```
int filectime (string nume_fisier)
```

```
int filegroup (string nume_fisier)
```

```
int fileinode (string nume_fisier)
```

```
int filemtime (string nume_fisier)
```

```
int fileowner (string nume_fisier)
```

```
int fileperms (string nume_fisier)
```

```
int filesize (string nume_fisier)
```

```
string filetype (string nume_fisier)
```

```
bool flock (int fp, int operatie [, int wouldblock])
```

```
int fopen (string nume_fisier, string mod [, int foloseste_cale_include])
```

```
int fpassthru (int fp)
```

```
int fputs (int fp, string sir [, int lungime])
```

```
string fread (int fp, int lungime)
```

```
mixed fscanf (int handle, string format [, string var1 ...])
```

Descriere

Modifică posesorul unui fișier.

Modifică zona cache cu statisticile fișierului.

Copiază un fișier.

Returnează componenta nume de catalog a unei căi.

Returnează spațiul disponibil dintr-un catalog.

Închide un pointer de fișier deschis.

Testează un pointer de fișier pentru a detecta sfârșitul fișierului.

Transferă într-un fișier datele de ieșire suspendate.

Preia un caracter dintr-un fișier.

Preia o linie dintr-un fișier și analizează linia în căutarea câmpurilor CSV.

Preia o linie dintr-un fișier.

Preia o linie dintr-un fișier și elimină etichetele HTML.

Citește un întreg fișier într-un tablou.

Verifică existența unui fișier.

Preia ora ultimei operații de acces la fișier.

Obține timpul de modificare al i-nodului pentru un fișier.

Obține grupul posesor al fișierului.

Obține i-nodul unui fișier.

Obține ora modificării fișierului.

Obține posesorul fișierului.

Obține permisiunile asociate unui fișier.

Obține dimensiunea unui fișier.

Obține tipul unui fișier.

Blochează un fișier.

Deschide un fișier sau o adresă URL.

Transcrie toate datele rămase într-un fișier.

Scrie într-un fișier.

Citește în condiții de siguranță date binare dintr-un fișier.

Analizează datele de intrare dintr-un fișier în conformitate cu un format.

Prototipul funcției

```

int fseek (int fp, int offset [, int baza])
array fstat (int fp)

int ftell (int fp)
int ftruncate (int fp, int dimensiune)

int fwrite (int fp, string sir
[, int lungime])
bool is_dir (string nume_fisier)

bool is_executable (string nume_fisier)
bool is_file (string nume_fisier)
bool is_link (string nume_fisier)

bool is_readable (string nume_fisier)
bool is_uploaded_file (string nume_fisier)

bool is_writable (string nume_fisier)
bool is_writeable (string nume_fisier)

int link (string tinta, string legatura)
int linkinfo (string cale)
array lstat (string nume_fisier)

int mkdir (string nume_cale, int mod)
bool move_uploaded_file (string
nume_fisier, string destinatie)
array pathinfo (string cale)
int fclose (int fp)
int fopen (string comanda, string mod)
int readfile (string nume_fisier
[, int utilizeaza_cale_include])
string readlink (string cale)
string realpath (string cale)
int rename (string nume_vechi,
string nume_nou)
int rewind (int fp)

int rmdir (string nume_catalog)
int set_file_buffer (int fp, int buffer)

```

Descriere

Caută într-o poziție specificată din interiorul fișierului.

Obține informații despre un fișier folosind un pointer al unui fișier deschis.

Obține poziția curentă în interiorul unui fișier.

Trunchiază un fișier pentru a-l aduce la lungimea specificată.

Scrie în condiții de siguranță date într-un fișier binar.

Determină dacă un nume de fișier se referă la un catalog.

Determină dacă un fișier este executabil.

Determină dacă fișierul este un fișier obișnuit.

Determină dacă un nume de fișier face referire la o legătură simbolică.

Determină dacă un fișier poate fi citit.

Determină dacă un fișier a fost încărcat prin intermediul comenzii HTTP POST.

Determină dacă se poate scrie într-un fișier. Funcția este similară cu `is_writeable()`.

Determină dacă se poate scrie într-un fișier. Funcția este similară cu `is_writable()`.

Creează o legătură hard.

Obține informații despre o legătură.

Oferă informații despre un fișier sau despre o legătură simbolică.

Creează un catalog.

Mută un fișier încărcat într-o nouă locație.

Returnează informații despre o cale.

Închide un canal.

Deschide un canal.

Trimite un fișier la ieșire.

Returnează ținta unei legături simbolice.

Returnează un nume de cale absolut, canonic.

Modifică numele unui fișier.

Deplasează înapoi poziția unui pointer de fișier.

Elimină un catalog.

Configurează memorarea în buffer a unui fișier.

Prototipul funcției

```

array stat (string nume_fisier)
int symlink (string tinta, string legatura)
string tempnam (string catalog, string prefix)
int tmpfile ()
int touch (string nume_fisier [, int ora])
int umask (int masca)
int unlink (string nume_fisier)

```

Descriere

Oferă informații despre un fișier.

Creează o legătură simbolică.

Creează un nume de fișier unic.

Creează un fișier temporar.

Determină ora modificării unui fișier.

Modifică u-masca curentă.

Șterge un fișier.

Funcții FTP**Prototipul funcției**

```

int ftp_chdir (int ftp_stream,
string catalog)
int ftp_cdup (int ftp_stream)
int ftp_connect (string gazda [, int port])
int ftp_delete (int ftp_stream, string cale)
int ftp_fget (int ftp_stream, int fp,
string fisier_distant, int mod)
int ftp_fput (int ftp_stream, int fp,
string fisier_distant, int mod)
int ftp_get (int ftp_stream, string
fisier_local, string fisier_distant,
int mod)
int ftp_login (int ftp_stream, string
nume_utilizator, string parola)
int ftp_mdtm (int ftp_stream, string
fisier_distant)
string ftp_mkdir (int ftp_stream,
string catalog)
array ftp_nlist (int ftp_stream,
string catalog)
int ftp_pasv (int ftp_stream, int pasv)
int ftp_put (int ftp_stream, string
fisier_distant, string fisier_local,
int mod)
string ftp_pwd (int ftp_stream)
int ftp_quit (int ftp_stream)
array ftp_rawlist (int ftp_stream,
string catalog)
int ftp_rename (int ftp_stream, string
initial, string final)
int ftp_rmdir (int ftp_stream, string
catalog)

```

Descriere

Comută în catalogul specificat dintr-un server FTP.

Comută în catalogul părinte.

Deschide o conexiune FTP.

Șterge un fișier din serverul FTP.

Descarcă un fișier din serverul FTP și salvează datele într-un fișier deschis.

Încarcă datele dintr-un fișier deschis în serverul FTP.

Descarcă un fișier din serverul FTP.

Deschide sesiunea de lucru cu o conexiune FTP deschisă.

Returnează ora ultimei modificări a fișierului specificat.

Creează un catalog pe serverul FTP.

Returnează o listă cu fișierele din catalogul specificat.

Activează sau dezactivează modul pasiv.

Încarcă un fișier în serverul FTP.

Returnează numele catalogului curent.

Închide o conexiune FTP.

Returnează o listă detaliată a fișierelor din catalogul specificat.

Modifică numele unui fișier din serverul FTP.

Elimină un catalog din serverul FTP.

Prototipul funcției

```
int ftp_site (int ftp_stream, string comanda)
int ftp_size (int ftp_stream, string
fisier_distant)
string ftp_systype (int ftp_stream)
```

Descriere

Trimite o comandă SITE serverului FTP.
 Returnează dimensiunea fișierului specificat.
 Returnează identificatorul tipului de sistem al serverului FTP aflat la distanță.

Funcții HTTP**Prototipul funcției**

```
int header (string sir)
bool headers_sent ()

int setcookie (string nume [, string
valoare [, int expira [, string cale
[, string domeniu [, int sigur]]]])
```

Descriere

Trimite un antet HTTP brut.
 Returnează true dacă au fost trimise antetele HTTP.
 Trimite o variabilă cookie.

Funcții IMAP, POP3 și NNTP**Prototipul funcției**

```
string imap_8bit (string sir)

array imap_alerts ()

int imap_append (int imap_stream, string
mbox, string mesaj [, string indicatoare])
string imap_base64 (string text)

string imap_binary (string sir)

string imap_body (int imap_stream, int
numar_mesaj [, int indicatoare])
object imap_check (int imap_stream)
string imap_clearflag_full (int
imap_stream, string secventa, string
indicator, string optiuni)
int imap_close (int imap_stream,
[, int indicatoare])
int imap_createmailbox (int imap_stream,
string mbox)
int imap_delete (int imap_stream, int
numar_mesaj [, int indicatoare])
```

Descriere

Convertește un șir pe 8 biți într-un șir care poate fi afișat cu ghilimele.
 Returnează mesaje de avertizare IMAP care s-au produs în timpul solicitării acestei pagini sau de la reinițializarea stivei de avertismente.
 Atașează un mesaj șir la o cutie poștală specificată.
 Decodifică texte codificate cu algoritmul BASE64.
 Convertește un șir pe 8 biți într-un șir BASE64.
 Citește corpul unui mesaj.
 Examinează cutia poștală curentă.
 Elimină indicatoarele din mesaje.
 Închide un flux IMAP.
 Creează o nouă cutie poștală.
 Marchează în vederea ștergerii un mesaj din cutia poștală curentă.

Prototipul funcției

```
int imap_deletemailbox (int imap_stream,
string mbox)
array imap_errors ()

int imap_expunge (int imap_stream)

array imap_fetch_overview (int imap_stream,
string secventa [, int indicatoare])
string imap_fetchbody (int imap_stream,
int numar_mesaj, string numar_parte
[, int indicatoare])
string imap_fetchheader (int imap_stream,
int nr_mesaj, int indicatoare)
object imap_fetchstructure (int imap_stream,
int nr_mesaj [, int indicatoare])
array imap_get_quota (int imap_stream,
string cota_radacina)
array imap_getmailboxes (int imap_stream,
string ref, string model)
array imap_getsubscribed (int imap_stream,
string ref, string model)
object imap_header (int imap_stream, int
nr_mesaj [, int explungime [, int
lungime_subiect [, string
gazda_prestabilita]]])
object imap_headerinfo (int imap_stream,
int nr_mesaj [, int explungime
[, int lungime_subiect
[, string gazda_prestabilita]]])
array imap_headers (int imap_stream)

string imap_last_error ()

array imap_listmailbox (int imap_stream,
string ref, string model)
array imap_listsubscribed (int
imap_stream, string ref, string model)
string imap_mail (string destinatar,
string subiect, string mesaj [, string
antete_suplimentare [, string cc
[, string bcc [, string roale]]]])
string imap_mail_compose (array
anvelopa, array corp)
```

Descriere

Șterge o cutie poștală.
 Returnează toate erorile IMAP care s-au produs de la acea cerere de pagină sau de la reinițializarea stivei de erori.
 Șterge toate mesajele marcate în vederea ștergerii.
 Citește o trecere în revistă a informațiilor incluse în antetele mesajului specificat.
 Preia secțiunea specificată din corpul unui mesaj.
 Returnează antetul unui mesaj.
 Citește structura unui anumit mesaj.
 Regăsește parametrii cotelor unei cutii poștale.
 Citește lista cutiilor poștale.
 Afișează lista tuturor cutiilor poștale la care s-a subscris.
 Citește antetul unui mesaj.
 Citește antetul unui mesaj.
 Returnează antetele tuturor mesajelor dintr-o cutie poștală.
 Returnează ultima eroare IMAP care s-a produs în timpul acestei cereri de pagină.
 Citește lista cutiilor poștale.
 Afișează toate cutiile poștale la care s-a subscris.
 Expediază un mesaj de e-mail.
 Creează un mesaj MIME pornind de la un corp și o anvelopă specificate.

Prototipul funcției

```
int imap_mail_copy (int imap_stream,
string lista_mesaje, string mbox
[, int indicatoare])
object imap_mailboxmsginfo (int
imap_stream)
array imap_mime_header_decode (string text)
int imap_msgno (int imap_stream, int uid)

int imap_num_msg (int imap_stream)

int imap_num_recent (int imap_stream)

int imap_open (string cutie_postala,
string nume_utilizator, string parola
[, int indicatoare])
int imap_ping (int imap_stream)
string imap_qprint (string sir)

int imap_renamemailbox (int imap_stream,
string cutie_veche, string cutie_noua)
int imap_reopen (int imap_stream, string
cutie_postala [, string indicatoare])
array imap_rfc822_parse_adrlist (string
adresa, string gazda_prestabilita)
array imap_rfc822_parse_headers (string
antete [, string gazda_prestabilita])
string imap_rfc822_write_address
(string cutie_postala, string gazda,
string personal)
array imap_scanmailbox (int imap_stream,
string ref, string model, string continut)
array imap_search (int imap_stream,
string criterii, int indicatoare)
int imap_set_quota (int imap_stream,
string cota_radacina, int cota_limita)
string imap_setflag_full (int imap_stream,
string secventa, string indicator,
string optiuni)
array imap_sort (int imap_stream, int
criterii, int invers, int optiuni)
object imap_status (int imap_stream,
string cutie_postala, int optiuni)
int imap_subscribe (int imap_stream,
string mbox)
```

Descriere

Copiază într-o cutie poștală mesajele specificate.

Obține informații despre cutia poștală curentă.

Decodifică elementele unui antet MIME.

Returnează numărul de secvență al mesajului pentru identificatorul unic dat.

Returnează numărul mesajelor din cutia poștală curentă.

Returnează numărul mesajelor recente din cutia poștală curentă.

Deschide un flux IMAP pentru o cutie poștală.

Verifică dacă un flux IMAP este activ.

Convertește un șir care poate fi afișat cu ghilimele într-un șir pe 8 biți.

Modifică numele unei cutii poștale.

Redeschide un flux IMAP.

Decelează o adresă RFC 822.

Decelează antete de poștă RFC 822.

Returnează o adresă de e-mail formatată în mod corespunzător.

Caută texte în cutiile poștale.

Caută în cutiile poștale mesaje care satisfac criteriile specificate.

Stabilește cotele pentru o cutie poștală.

Stabilește indicatoarele unui mesaj.

Sortează un tablou cu antete de mesaj.

Returnează informații de stare referitoare la o cutie poștală.

Se abonează la o cutie poștală.

Prototipul funcției

```
int imap_uid (int imap_stream, int
nr_mesaj)
int imap_undelete (int imap_stream,
int numar_mesaj)
int imap_unsubscribe (int imap_stream,
string mbox)
string imap_utf7_decode (string text)

string imap_utf7_encode (string data)

string imap_utf8 (string text)
```

Descriere

Returnează identificatorul unic (UID) pentru numărul de secvență dat al mesajului.

Reinițializează indicatorul de ștergere asociat unui mesaj.

Anulează abonamentul la o cutie poștală.

Decodifică un șir codificat cu algoritmul UTF-7 modificat.

Convertește date pe 8 biți în text codificat cu algoritmul UTF-7 modificat.

Convertește text la UTF-8.

Funcții de poștă**Prototipul funcției**

```
bool mail (string destinatar, string
subiect, string mesaj [, string
antete_suplimentare [, string
[parametri_suplimentari]]])
int ezmlm_hash (string adresa)
```

Descriere

Expediază mesaje poștale.

Calculează valoarea hash necesară EZMLM.

Funcții matematice**Prototipul funcției**

```
mixed abs (mixed numar)

float acos (float argument)

float asin (float argument)

float atan (float argument)

float atan2 (float y, float x)

string base_convert (string numar int
din_baza, int in_baza)
int bindec (string sir_binar)
float ceil (float valoare)

float cos (float argument)
string decbin (int numar)
string dechex (int numar)
```

Descriere

Returnează valoarea absolută a argumentului specificat.

Returnează valoarea arccosinusului argumentului specificat.

Returnează valoarea arcsinusului argumentului specificat.

Returnează valoarea arctangentei argumentului specificat.

Returnează valoarea arctangentei punctului specificat.

Convertește un număr dintr-o bază arbitrară în alta.

Convertește o valoare binară în baza 10.

Rotunjește o valoare la cel mai apropiat întreg mai mare sau egal cu valoarea.

Returnează cosinusul valorii specificate.

Convertește o valoare din baza 10 în binar.

Convertește o valoare din baza 10 în hexazecimal.

Prototipul funcției

```

string decoct (int numar)
double deg2rad (double numar)
float exp (float argument)

float floor (float valoare)

int getrandmax ()

int hexdec (string sir_hexa)

double lcg_value ()

float log (float argument)

float log10 (float argument)

mixed max (mixed argument1, mixed
argument2, mixed argumentn)
mixed min (mixed argument1, mixed
argument2, mixed argumentn)
int mt_rand ([int minim [, int maxim]])

void mt_srand (int seed)

int mt_getrandmax ()

string number_format (float numar, int
zecimale, string punct_zecimal, string
separator_mii)
int octdec (string sir_octal)
double pi ()
float pow (float baza, float exponent)

double rad2deg (double numar)
int rand ([int minim [, int maxim]])

double round (double valoare
[, int precizie])
float sin (float argument)
float sqrt (float argument)

void srand (int seed)

```

Descriere

Convertește o valoare din baza 10 în octal.
 Convertește un unghi din grade în radiani.
 Returnează *e* (baza algoritmilor naturali) ridicat la puterea specificată.
 Rotunjește o valoare la cel mai apropiat întreg mai mic sau egal cu valoarea.
 Returnează cea mai mare valoare posibilă care poate fi returnată de funcția `rand ()`.
 Convertește o valoare hexazecimală în baza 10.
 Returnează o valoare aleatoare folosind un generator congruențial liniar combinat.
 Returnează logaritmul natural al valorii specificate.
 Returnează logaritmul în bază 10 al valorii specificate.
 Returnează cea mai mare valoare din cele specificate.
 Returnează cea mai mică valoare din cele specificate.
 Returnează o valoare aleatoare din interiorul unui domeniu specificat.
 Inițiază generatorul automat de numere folosit de funcția `mt_rand ()`.
 Returnează cea mai mare valoare posibilă pe care o poate returna funcția `mt_rand ()`.
 Formatează un număr prin gruparea miilor.
 Convertește o valoare din octal în baza 10.
 Returnează valoarea numărului π .
 Returnează rezultatul ridicării valorii specificate la puterea specificată.
 Convertește un unghi din radiani în grade.
 Returnează o valoare aleatoare din domeniul specificat.
 Rotunjește o valoare cu virgulă mobilă.
 Returnează sinusul valorii specificate.
 Returnează rădăcina pătrată a valorii specificate.
 Inițiază generatorul de numere aleatoare folosit de funcția `rand ()`.

Prototipul funcției

```
float tan (float argument)
```

Descriere

Returnează tangenta valorii specificate.

Funcții diverse**Prototipul funcției**

```

int connection_aborted ()

int connection_status ()
mixed constant (string nume)
int define (string nume, mixed valoare
[, int caz_insensibil])
int defined (string nume)
void die (string mesaj)

mixed eval (string cod_sir)
void exit (mixed stare)
object get_browser ([string
agent_utilizator])
bool highlight_file (string nume_fisier)

bool highlight_string (string sir)

int ignore_user_abort ([int parametru])

array iptcparse (string iptcbloc)

void leak (int octeti)

string pack (string format
[, mixed argumente ...])
bool show_source (string nume_fisier)

void sleep (int secunde)

int uniqid (string prefix [, bool log])

array unpack (string format, string data)
void usleep (int microsecunde)

```

Descriere

Returnează `true` în cazul în care clientul a fost deconectat.
 Returnează starea conexiunii curente.
 Returnează valoarea unei constante.
 Definește o constantă denumită.
 Verifică dacă o constantă denumită există.
 Afișează la ieșire un mesaj și încheie execuția scriptului curent.
 Evaluează un șir sub formă de cod PHP.
 Încheie execuția scriptului curent.
 Determină posibilitățile browserului client.
 Afișează un fișier cu elementele de sintaxă evidențiate.
 Afișează un șir cu elementele de sintaxă evidențiate.
 Specifică dacă prin deconectarea unui client trebuie să se abandoneze execuția scriptului.
 Decelează un bloc IPTC (International Press Telecommunications Council) în etichete singulare.
 Provoacă scurgeri de memorie (funcția este utilă pentru testarea programelor).
 Împachetează datele într-un fișier binar.
 Afișează un fișier cu elementele de sintaxă evidențiate.
 Amână execuția pentru un interval de timp specificat.
 Generează un identificator unic din punct de vedere statistic.
 Despachetează datele dintr-un șir binar.
 Amână execuția pentru un interval de timp specificat în microsecunde.

Funcții MySQL

Prototipul funcției

```
int mysql_affected_rows ([resource
identificator_legatura])

int mysql_change_user (string utilizator,
string parola [, string baza_de_date
[, resource identificator_legatura]])

bool mysql_close ([resource
identificator_legatura])

resource mysql_connect ([string numegazda
[:port] [:cale/spre/soclu] [, string
nume_utilizator [, string parola]])

int mysql_create_db (string nume_baza_de_
date [, resource identificator_legatura])

bool mysql_data_seek (resource
identificator_rezultat, int numar_rand)

string mysql_db_name (resource rezultat,
int rand [, mixed camp])

resource mysql_db_query (string
baza_de_date, string interogare
[, resource identificator_legatura])

bool mysql_drop_db (string nume_baza_de_
date [, resource identificator_legatura])

int mysql_errno ([resource
identificator_legatura])

string mysql_error ([resource
identificator_legatura])

string mysql_escape_string (string
sir_fara_escape)

array mysql_fetch_array (resource
rezultat [, int tip_rezultat])

array mysql_fetch_assoc (resource
rezultat)

object mysql_fetch_field (resource
rezultat [, int offset_camp])

array mysql_fetch_lengths (resource
rezultat)

object mysql_fetch_object (resource
rezultat [, int tip_rezultat])

array mysql_fetch_row (resource rezultat)

string mysql_field_flags (resource
rezultat, int offset_camp)
```

Descriere

Returnează numărul rândurilor afectate în operația MySQL anterioară.

Returnează identitatea utilizatorului asociată conexiunii active.

Închide o conexiune MySQL.

Deschide o conexiune cu un server MySQL.

Creează o bază de date MySQL.

Mută pointerul intern al setului de rezultate.

Returnează datele din setul de rezultate.

Trimite o interogare MySQL în vederea execuției.

Șterge o bază de date MySQL.

Returnează valoarea numerică a mesajului de eroare provenit de la operația MySQL anterioară.

Returnează textul mesajului de eroare provenit de la operația MySQL anterioară.

Inserează într-un șir caractere escape în vederea utilizării șirului într-o interogare.

Preia un rând din rezultat sub forma unui tablou asociativ, a unui tablou numeric sau ambele.

Preia un rând din rezultat sub forma unui tablou asociativ.

Returnează informații despre coloanele dintr-un set de rezultate sub forma unui obiect.

Returnează lungimea fiecărei coloane dintr-un set de rezultate.

Preia un rând din rezultat sub forma unui obiect.

Returnează un rând din rezultat sub forma unui tablou enumerat.

Returnează indicatoarele asociate unui câmp specificat dintr-un set de rezultate.

Prototipul funcției

```
int mysql_field_len (resource rezultat,
int offset_camp)

string mysql_field_name (resource
rezultat, int index_camp)

int mysql_field_seek (resource rezultat,
int offset_camp)

string mysql_field_table (resource
rezultat, int offset_camp)

string mysql_field_type (resource
rezultat, int offset_camp)

int mysql_free_result (resource rezultat)

string mysql_get_client_info (void)

string mysql_get_host_info ([resource
rezultat])

int mysql_get_proto_info ([resource
identificator_legatura])

int mysql_get_server_info ([resource
identificator_legatura])

int mysql_insert_id ([resource
identificator_legatura])

resource mysql_list_dbs ([resource
identificator_legatura])

resource mysql_list_fields (string
nume_baza_de_date, string nume_tabel
[, resource identificator_legatura])

resource mysql_list_tables (string
baza_de_date [, resource
identificator_legatura])

int mysql_num_fields (resource rezultat)

int mysql_num_rows (resource rezultat)

resource mysql_pconnect ([string
nume_gazda [:port] [:cale/spre/soclu]
[, string nume_utilizator
[, string parola]])

resource mysql_query (string interogare
[, resource identificator_legatura])

mixed mysql_result (resource rezultat,
int rand [, mixed camp])

bool mysql_select_db (string nume_baza_de_
date [, resource identificator_legatura])

string mysql_tablename (resource rezultat,
int i)
```

Descriere

Returnează lungimea câmpului specificat dintr-un set de rezultate.

Returnează numele câmpului specificat dintr-un set de rezultate.

Poziționează pointerul rezultatului la un anumit offset al câmpului.

Returnează numele tabelului de unde a provenit câmpul specificat din setul de rezultate.

Returnează tipul câmpului specificat din setul de rezultate.

Eliberează memoria alocată setului de rezultate.

Returnează informații despre clientul MySQL.

Returnează informații despre gazda MySQL.

Returnează informații despre protocolul MySQL.

Returnează informații despre serverul MySQL.

Returnează identificatorul generat din inserția anterioară a unui rând care conține un câmp AUTO_INCREMENT.

Afișează bazele de date disponibile dintr-un server MySQL.

Afișează coloanele unui tabel MySQL.

Afișează tabelele dintr-o bază de date MySQL.

Returnează numărul coloanelor dintr-un set de rezultate.

Returnează numărul rândurilor dintr-un set de rezultate.

Deschide o conexiune persistentă cu un server MySQL.

Trimite o interogare MySQL în vederea execuției.

Returnează datele dintr-un set de rezultate.

Selectează o bază de date MySQL.

Returnează numele tabelului care conține câmpul din setul de rezultate.

Funcții de control al datelor de ieșire

Prototipul funcției

`void ob_end_clean ()`

`void ob_end_flush ()`

`void flush ()`

`string ob_get_contents ()`

`string ob_get_length ()`

`string ob_gzhandler (string buffer)`

`void ob_implicit_flush ([int indicator])`

`void ob_start ([string apelinvers_iesire])`

Descriere

Șterge conținutul bufferului de ieșire și dezactivează memorarea în buffer a datelor de ieșire.

Golește bufferul de ieșire și dezactivează memorarea datelor de ieșire în buffer.

Golește bufferul de ieșire.

Returnează conținutul bufferului de ieșire.

Returnează lungimea bufferului de ieșire.

O funcție cu apel invers pentru arhivarea (cu programul gzip) a conținutului bufferului de ieșire. (Activată prin intermediul unei opțiuni din `php.ini`.)

Activează/dezactivează golirea implicită.

Activează memorarea în buffer a datelor de ieșire.

Opțiuni și informații PHP

Prototipul funcției

`int assert (string|bool asertiune)`

`mixed assert_options (int entitate [, mixed valoare])`

`int dl (string biblioteca)`

`bool extension_loaded (string nume)`

`string get_cfg_var (string nume_variabila)`

`string get_current_user ()`

`array get_defined_constants ()`

`array get_extension_funcs (string nume_modul)`

`array get_included_files ()`

`array get_loaded_extensions ()`

`long get_magic_quotes_gpc ()`

`long get_magic_quotes_runtime ()`

Descriere

Verifică dacă o aserțiune este falsă.

Stabilește sau obține diferite indicatoare de aserțiune.

Încarcă o extensie PHP.

Determină dacă a fost încărcată o extensie.

Obține valoarea unei opțiuni de configurare PHP.

Obține numele posesorului scriptului PHP curent.

Returnează un tablou asociativ care conține numele și valoarea fiecărei constante definite.

Returnează un tablou care conține numele funcțiilor din cadrul unui modul.

Returnează un tablou care conține numele fișierelor la care un script a obținut accesul folosind `include_once`.

Returnează un tablou care conține numele tuturor modulelor încărcate.

Obține valoarea curentă a parametrului `magic_quotes_gpc`.

Obține valoarea curentă a parametrului `magic_quotes_runtime`.

Prototipul funcției

`array get_required_files ()`

`string getenv (string nume_variabila)`

`int getlastmod ()`

`int getmyinode ()`

`int getmypid ()`

`int getmyuid ()`

`array getrusage ([int cine])`

`string ini_alter (string nume_variabila, string valoare_noua)`

`string ini_get (string nume_variabila)`

`string ini_restore (string nume_variabila)`

`string ini_set (string nume_variabila, string valoare_noua)`

`string php_logo_guid ()`

`string php_sapi_name ()`

`string php_uname ()`

`void phpcredits (int indicator)`

`int phpinfo ([int entitate])`

`string phpversion ()`

`void putenv (string parametru)`

`logic set_magic_quotes_runtime (int configurare_noua)`

`void set_time_limit (int secunde)`

`string zend_logo_guid ()`

`string zend_version ()`

Descriere

Returnează un tablou care conține numele fișierelor la care un script a obținut accesul folosind `require_once` sau `include_once`.

Obține valoarea unei variabile de mediu.

Obține ora ultimei modificări a paginii curente.

Obține numărul i-nod al scriptului curent.

Obține identificatorul de proces al scriptului PHP.

Obține identificatorul unic (UID) al posesorului scriptului curent.

Obține informațiile privind gradul curent de utilizare a resurselor.

Stabilește valoarea unei opțiuni de configurare.

Obține valoarea unei opțiuni de configurare.

Restaurează valoarea unei opțiuni de configurare.

Stabilește valoarea unei opțiuni de configurare.

Obține imaginea siglei PHP.

Determină tipul de interfață dintre serverul Web și PHP.

Returnează informații despre sistemul de operare sub care a fost construit PHP.

Afișează colectivul care a creat limbajul PHP.

Afișează informații despre configurația și starea PHP.

Obține versiunea PHP curentă.

Stabilește valoarea unei variabile de mediu.

Stabilește valoarea curentă a parametrului `magic_quotes_runtime`.

Stabilește limita maximă a timpului de execuție.

Obține imaginea siglei Zend.

Obține numărul versiunii motorului Zend curent.

Funcții de execuție a programelor

Prototipul funcției

`string escapeshellarg (string argument)`

`string escapeshellcmd (string comanda)`

Descriere

Inserează într-un șir caracterele escape adecvate, în vederea utilizării șirului ca argument pentru interfața de tip shell.

Convertește metacaracterele specifice interfeței shell.

Prototipul funcției

```
string exec (string comanda [, string
tablou [, int var_returnata]])
void passthru (string comanda [, string
tablou [, int var_returnata]])
string system (string comanda [, int
var_returnata])
```

Descriere

Execută un program extern.

Execută un program extern și afișează datele de ieșire în formă brută.

Execută un program extern și afișează datele de ieșire.

Funcții POSIX

Prototipul funcției

```
string posix_ctermid ()
string posix_getcwd ()
int posix_getegid ()
int posix_geteuid ()
int posix_getgid ()
array posix_getgrgid (int gid)
array posix_getgrnam (string nume)
array posix_getgroups ()
int posix_getid (int pid)
string posix_getlogin ()
int posix_getpgid (int pid)
int posix_getpgrp ()
int posix_getppid ()
array posix_getpwnam (string
nume_utilizator)
array posix_getpwuid (int uid)
array posix_getrlimit ()
int posix_getuid ()
```

Descriere

Obține numele căii terminalului de control asociat procesului curent.

Returnează numele căii catalogului curent.

Returnează identificatorul de grup efectiv al procesului curent.

Returnează identificatorul de utilizator efectiv al procesului curent.

Returnează identificatorul de grup real al procesului curent.

Returnează informații despre grupul de procese specificat.

Returnează informații despre grupul de procese specificat.

Returnează setul de grupuri al procesului curent.

Obține identificatorul SID al procesului specificat.

Returnează numele de deschidere a sesiunii de lucru al utilizatorului posesor al procesului curent.

Obține identificatorul grupului de procese din care face parte procesul specificat.

Returnează identificatorul de grup al procesului curent.

Returnează identificatorul de proces al procesului părinte.

Returnează informații despre utilizatorul specificat.

Returnează informații despre utilizatorul specificat.

Returnează informații despre limitele resurselor sistemului.

Returnează identificatorul de utilizator real al procesului curent.

Prototipul funcției

```
bool posix_isatty (int fd)
bool posix_mkfifo (string nume_cale,
int mod)
bool posix_setgid (int gid)
int posix_setpgid (int pid, int pgid)
int posix_setsid ()
bool posix_setuid (int uid)
array posix_times ()
string posix_tty_name (int fd)
array posix_uname ()
```

Descriere

Determină dacă un descriptor de fișier este asociat unui terminal interactiv.

Creează un fișier FIFO special, în speță un canal cu nume.

Stabilește identificatorul GID efectiv al procesului curent.

Stabilește identificatorul grupului de procese din care face parte procesul respectiv.

Transformă procesul curent într-un proces lider de sesiune.

Stabilește identificatorul UID efectiv al procesului curent.

Obține informațiile privind utilizarea CPU de către procesul curent.

Determină numele dispozitivului terminal.

Obține numele sistemului.

Funcții POSIX pentru manipularea expresiilor regulate extinse

Prototipul funcției

```
int ereg (string model, string sir
[, array regs])
string ereg_replace (string model, string
inlocuitor, string sir)
int eregi (string model, string sir
[, array regs])
string eregi_replace (string model, string
inlocuitor, string sir)
array split (string model, string sir,
[, int limita])
array spliti (string model, string sir,
[, int limita])
string sql_regcase (string sir)
```

Descriere

Determină dacă un șir corespunde unei expresii regulate specificate.

Înlocuiește un sub-șir care corespunde unei expresii regulate.

Determină dacă un șir corespunde unei expresii regulate specificate (funcția este insensibilă la diferența dintre majuscule și minuscule).

Înlocuiește un sub-șir care corespunde unei expresii regulate (funcția este insensibilă la diferența dintre majuscule și minuscule).

Folosește o expresie regulată pentru a diviza un șir într-un tablou.

Folosește o expresie regulată pentru a diviza un șir într-un tablou (funcția este insensibilă la diferența dintre majuscule și minuscule).

Returnează un model insensibil la diferența între majuscule și minuscule, bazat pe o expresie regulată specificată.

Funcții de manipulare a sesiunilor

Prototipul funcției

`string session_cache_limiter ([string limitator_cache])`

`bool session_decode (string date)`

`bool session_destroy ()`

`string session_encode ()`

`array session_get_cookie_params ()`

`string session_id ([string id])`

`bool session_is_registered (string nume)`

`string session_module_name ([string modul])`

`string session_name ([string nume])`

`bool session_register (mixed nume [, mixed ...])`

`string session_save_path ([string cale])`

`void session_set_cookie_params (int durata_de_viata [, string cale [, string domeniu]])`

`void session_set_save_handler (string deschide, string inchide, string citește, string scrie, string distruge, string gc)`

`bool session_start ()`

`bool session_unregister (string nume)`

`void session_unset ()`

Descriere

Obține sau stabilește opțiunea curentă de limitare a memoriei cache.

Decodifică datele aferente sesiunii.

Distruge datele înregistrate pentru o sesiune.

Codifică datele sesiunii curente.

Obține parametrii variabilelor cookie ale sesiunii.

Obține sau stabilește identificatorul sesiunii curente.

Determină dacă o anumită variabilă este înregistrată într-o sesiune.

Obține sau stabilește modulul sesiunii curente.

Obține sau stabilește numele sesiunii curente.

Înregistrează una sau mai multe variabile în cadrul sesiunii curente.

Obține sau stabilește calea de salvare a sesiunii curente.

Stabilește parametrii variabilelor cookie ale sesiunii.

Stabilește funcțiile de stocare la nivel de utilizator ale sesiunii.

Inițializează datele sesiunii.

Anulează înregistrarea unei variabile în sesiunea curentă.

Eliberează memoria alocată tuturor variabilelor sesiunii.

Funcții cu șiruri

Prototipul funcției

`string addslashes (string sir, string lista_caractere)`

`string addslashes (string sir)`

`string bin2hex (string sir)`

`string chop (string sir)`

`string chr (int ascii)`

Descriere

Delimitează un șir folosind caractere slash, în conformitate cu o listă specificată de caractere ce vor fi utilizate drept ghilimele.

Încadrează un șir între caractere slash.

Convertește datele binare în hexazecimal.

Elimină spațiile albe de final.

Returnează un caracter cu o valoare ASCII specificată.

Prototipul funcției

`string chunk_split (string sir [, int lungime_bucata [, string sfarsit]])`

`string convert_cyr_string (string sir, string din, string in)`

`mixed count_chars (string sir, [, int mod])`

`int crc32 (string sir)`

`string crypt (string sir [, string sare])`

`echo (string argument1, string [argumentn] ...)`

`array explode (string separator, string sir [, int limita])`

`string get_html_translation_table (int tabel [int stil_citare])`

`array get_meta_tags (string nume_fisier [int use_include_path])`

`string hebreu (string text_iudaic [, int max_caractere_pe_linie])`

`string hebreuc (string text_iudaic [, int max_caractere_pe_linie])`

`string htmlentities (string sir [, int stil_citare])`

`string htmlspecialchars (string sir, [, int stil_citare])`

`string implode (string clei, array piese)`

`string join (string clei, array piese)`

`int levenshtein (string sir1, string sir2)` Calculează distanța Levenshtein între două șiruri.

`int levenshtein (string sir1, string sir2, int cost_ins, int cost_rep, int cost_del)` Calculează distanța Levenshtein între două șiruri.

`int levenshtein (string sir1, string sir2, function cost)` Calculează distanța Levenshtein între două șiruri.

`array localeconv ()` Obține informațiile de formatare numerică pentru locale curentă.

`string ltrim (string sir)` Elimină spațiile albe de la începutul unui șir.

`string md5 (string sir)` Calculează valoarea hash MD5 a unui șir.

`string metaphone (string sir)` Calculează valoarea metafonică a unui șir.

`string n12br (string sir)` Inserează salturi de linie HTML anterior tuturor caracterelor linie nouă dintr-un șir.

Descriere

Divizează un șir în sub-șiruri.

Convertește un set de caractere chirilice într-un alt set de caractere chirilice.

Returnează informații despre caracterele folosite într-un șir.

Calculează valoarea polinomului CRC32 al unui șir.

Criptează un șir folosind algoritmul DES.

Afișează unul sau mai multe șiruri.

Divizează un șir în sub-șiruri, folosind un șir separator specificat.

Returnează tabelul de conversie folosit de funcțiile `htmlspecialchars ()` și `htmlentities ()`.

Extrage toate atributele conținutului etichetelor meta dintr-un fișier și returnează un tablou care conține aceste atribute.

Convertește un text logic scris cu caractere iudaice în text vizual.

Convertește un text logic scris cu caractere iudaice în text vizual, folosind conversia caracterului linie nouă.

Convertește toate caracterele posibile în entități HTML.

Convertește toate caracterele speciale în entități HTML.

Unește elementele unui tablou cu un șir.

Unește elementele unui tablou cu un șir.

Prototipul funcției

| | |
|--|--|
| <code>int ord (string sir)</code> | Descriere
Returnează valoarea ASCII a unui caracter. |
| <code>void parse_str (string sir [, array tablou])</code> | Defalcă un șir în variabile. |
| <code>print (string argument)</code> | Afișează un șir. |
| <code>int printf (string format [, mixed argumente ...])</code> | Afișează un șir formatat. |
| <code>string quoted_printable_decode (string sir)</code> | Convertește un șir care poate fi afișat cu ghilimele într-un șir pe 8 biți. |
| <code>string quotemeta (string sir)</code> | Convertește caracterele meta din cadrul unui șir. |
| <code>string rtrim (string sir)</code> | Elimină spațiile albe de final din cadrul unui șir. |
| <code>mixed sscanf (string sir, string format [, string var1 ...])</code> | Analizează datele de intrare dintr-un șir, în conformitate cu formatul specificat. |
| <code>string setlocale (mixed categorie, string locale)</code> | Stabilește informațiile despre locale. |
| <code>int similar_text (string primul, string al_doilea [, double procent])</code> | Calculează gradul de asemănare între două șiruri. |
| <code>string soundex (string sir)</code> | Calculează valoarea Soundex a unui șir. |
| <code>string sprintf (string format [, mixed argumente...])</code> | Returnează un șir formatat. |
| <code>int strncasecmp (string sir1, string sir2, int n)</code> | Compară primele <i>n</i> caractere a două șiruri (funcția este insensibilă la diferența dintre majuscule și minuscule). |
| <code>int strcasecmp (string sir1, string sir2)</code> | Compară două șiruri (funcția este insensibilă la diferența dintre majuscule și minuscule). |
| <code>string strchr (string car_cu_fan, string ac)</code> | Găsește prima apariție a unui caracter specificat în cadrul unui șir. |
| <code>int strcmp (string sir1, string sir2)</code> | Compară două șiruri. |
| <code>int strcoll (string sir1, string sir2)</code> | Compară două șiruri, bazându-se pe informațiile locale curente. |
| <code>string strcspn (string sir1, string sir2)</code> | Determină lungimea sub-șirului inițial care nu corespunde unui șir specificat. |
| <code>string strip_tags (string sir [, string etichete_permise])</code> | Elimină etichetele HTML și PHP dintr-un șir. |
| <code>string stripslashes (string sir)</code> | Elimină caracterele slash dintr-un șir cu formatul returnat de funcția addslashes (). |
| <code>string stripslashes (string sir)</code> | Elimină caracterele slash dintr-un șir cu formatul returnat de funcția addslashes (). |
| <code>string strpos (string car_cu_fan, string ac)</code> | Găsește prima apariție a unui sub-șir specificat în cadrul unui șir (funcția este insensibilă la diferența dintre majuscule și minuscule). |
| <code>int strlen (string sir)</code> | Obține lungimea unui șir. |

Prototipul funcției

| | |
|---|--|
| <code>int strnatcmp (string sir1, string sir2)</code> | Descriere
Compară două șiruri folosind un argument de ordonare naturală. |
| <code>int strnatcasecmp (string sir1, string sir2)</code> | Compară două șiruri folosind un argument de ordonare naturală (funcția este insensibilă la diferența dintre majuscule și minuscule). |
| <code>int strncmp (string sir1, string sir2, int n)</code> | Compară primele <i>n</i> caractere a două șiruri. |
| <code>string str_pad (string intrare, int lungime_completare [, string sir_completare [, int tip_completare]])</code> | Completează un șir până la o lungime specificată. |
| <code>int strpos (string car_cu_fan, string ac [, int offset])</code> | Găsește prima apariție a unui sub-șir specificat în cadrul unui șir. |
| <code>string strrchr (string car_cu_fan, string ac)</code> | Găsește ultima apariție a unui caracter din cadrul unui șir. |
| <code>string str_repeat (string intrare, int multiplicator)</code> | Returnează un șir care conține mai multe repetiții ale unui șir specificat. |
| <code>string strrev (string sir)</code> | Inversează ordinea unui șir. |
| <code>int strrpos (string car_cu_fan, string ac)</code> | Găsește ultima apariție a unui caracter specificat în cadrul unui șir. |
| <code>int strspn (string sir1, string sir2)</code> | Găsește lungimea sub-șirului inițial care corespunde unui șir dat. |
| <code>string strstr (string car_cu_fan, string ac)</code> | Găsește prima apariție a unui sub-șir. |
| <code>string strtok (string argument1, string argument2)</code> | Defalcă un șir în argumente. |
| <code>string strtolower (string sir)</code> | Convertește un șir la caractere minuscule. |
| <code>string strtoupper (string sir)</code> | Convertește un șir la caractere majuscule. |
| <code>mixed str_replace (mixed cauta, mixed inlocuire, mixed subiect)</code> | Înlocuiește toate aparițiile unui șir de căutare specificat cu un șir de înlocuire de asemenea specificat. |
| <code>string strtr (string sir, string din, string in)</code> | Convertește caracterele specificate. |
| <code>string substr (string sir, int inceput [, int lungime])</code> | Returnează un sub-șir, specificat prin poziție. |
| <code>int substr_count (string car_cu_fan, string ac)</code> | Numără aparițiile unui sub-șir. |
| <code>string substr_replace (string sir, string inlocuire, int inceput [, int lungime])</code> | Înlocuiește un sub-șir în funcție de poziție. |
| <code>string trim (string sir)</code> | Elimină caracterele de tip spațiu alb de la începutul, respectiv sfârșitul unui șir. |
| <code>string ucfirst (string sir)</code> | Convertește primul caracter al unui șir în majuscule. |
| <code>string ucwords (string sir)</code> | Convertește primul caracter al fiecărui cuvânt din cadrul unui șir la majuscule. |
| <code>string wordwrap (string sir [, int latime [, string break [, int taie]]])</code> | Înfășoară un șir pornind de la o lungime specificată, folosind un caracter specificat. |

Funcții URL

Prototipul funcției

`string base64_decode (string date_codificate)`

`string base64_encode (string date)`

`array parse_url (string url)`

`string rawurldecode (string sir)`

`string rawurlencode (string sir)`

`string urldecode (string sir)`

`string urlencode (string sir)`

Descriere

Decodifică date MIME BASE64.

Codifică date folosind MIME BASE64.

Analizează o adresă URL și returnează componentele sale.

Decodifică un șir codificat URL.

Codifică o adresă URL folosind metoda descrisă în documentul RFC 1738.

Decodifică un șir codificat URL.

Codifică un șir prin metoda URL.

Funcții cu variabile

Prototipul funcției

`double doubleval (mixed var)`

`bool empty (mixed var)`

`string gettype (mixed var)`

`array get_defined_vars ()`

`string get_resource_type (resource variabila)`

`int intval (mixed var [, int base])`

`bool is_array (mixed variabila)`

`bool is_bool (mixed variabila)`

`bool is_double (mixed variabila)`

`bool is_float (mixed variabila)`

`bool is_int (mixed variabila)`

`bool is_integer (mixed variabila)`

`bool is_long (mixed variabila)`

`bool is_null (mixed variabila)`

`bool is_numeric (mixed variabila)`

`bool is_object (mixed variabila)`

`bool is_real (mixed variabila)`

`bool is_resource (mixed variabila)`

`bool is_scalar (mixed variabila)`

Descriere

Obține o valoare de tip dublu corespunzătoare valorii unei variabile.

Determină dacă valoarea unei variabile este stabilită.

Obține tipul unei variabile.

Returnează un tablou care conține numele tuturor variabilelor definite.

Returnează un șir care reprezintă tipul resursei unei variabile care reprezintă o resursă.

Obține o valoare întreagă care corespunde valorii unei variabile.

Determină dacă o variabilă este un tablou.

Determină dacă o variabilă este de tip boolean.

Determină dacă o variabilă este de tip dublu.

Determină dacă o variabilă este de tip float.

Determină dacă o variabilă este de tip întreg.

Determină dacă o variabilă este de tip întreg.

Determină dacă o variabilă este de tip long.

Determină dacă valoarea unei variabile este null.

Determină dacă o variabilă este un număr sau un șir numeric.

Determină dacă o variabilă este de tip obiect.

Determină dacă o variabilă este de tip real.

Determină dacă o variabilă este o resursă.

Determină dacă o variabilă este de tip scalar.

Prototipul funcției

`bool is_string (mixed variabila)`

`int isset (mixed variabila)`

`void print_r (mixed expresie)`

`string serialize (mixed valoare)`

`bool settype (mixed var, string tip)`

`string strval (mixed var)`

`mixed unserialize (string sir)`

`void unset (mixed var [, mixed var [, ...]])`

`void var_dump (mixed expresie)`

Descriere

Determină dacă o variabilă este de tip șir.

Determină dacă a fost stabilită valoarea unei variabile.

Afișează informații despre o variabilă, inteligibile pentru operatorul uman.

Generează o reprezentare a unei valori care poate fi stocată.

Stabilește tipul unei variabile.

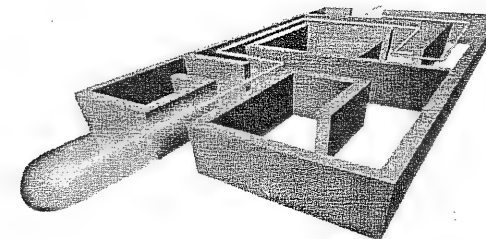
Obține o valoare șir care corespunde valorii unei variabile.

Creează o valoare PHP pornind de la o reprezentare stocată.

Anulează alocarea unei valori pentru o variabilă.

Afișează informații despre o variabilă.

Index



SIMBOLURI

- \$ (simbolul dolarului)
 - formarea numelor de variabilă PHP, 28
 - secvența escape PHP, 27-28
 - sesizarea diferenței între constante și variabile, 71
- \$\$ (o pereche de simboluri ale dolarului), 73
- % (simbolul procentului)
 - utilizarea ca operator modulo, 34
 - utilizarea ca procent, în loc de directivă, 144
 - utilizarea pentru directive, 142
- & (caracter ampersand), 114
- && (caracter ampersand dublu), 88
- * (asterisc)
 - ca operator de multiplicare, 33
 - ca valoare de repetiție, 153
- */ (asterisc, caracter slash orientat înainte), 18
- < (operator relațional), 86
- () (paranteze), 64
- { } (paranteze acolade)
 - identificarea variabilelor de șablon, 320
 - specificarea repetițiilor pentru expresiile regulate, 153
- <<- (operator de decrementare), 34, 34
- <<, utilizare cu etichetele HTML, 17
- <?, utilizare cu semnul egal, 64
- + (operator de adunare), 33, 153
- ++ (operator de incrementare), 34, 34
- +=, atribuire rapidă a operatorului +, 76
- , (virgulă)
 - instrucțiuni echo care folosesc, 64
 - specificarea argumentelor, 394
- (operator de scădere), 33
- . (punct), operator de concatenare
 - aliasuri pentru cataloage, 178
 - trecere în revistă, 34
 - unirea a două expresii șir, 62
- ... (puncte de suspensie)
 - prototipuri de funcții, 394
- / (caracter slash orientat înainte)
 - accesul la scripturile PHP via URL, 20
 - componentele căilor indicate prin, 365
 - scrierea instrucțiunilor PHP, 17
 - simbolizarea catalogului rădăcină, 177, 365
- / (operator de împărțire), 33
- /* (caracter slash orientat înainte, asterisc), 18
- // (două caractere slash orientate înainte), 17
- /xnn (număr în hexazecimal), 140
- /xxx (număr în octal), 140
- ; (punct și virgulă), 29, 62
- = (semnul egal)
 - abreviere pentru echo, 64
 - comparație între instrucțiunile de atribuire și ecuațiile matematice, 30
 - instrucțiuni de atribuire, 29
- =>, asocierea cheilor cu valori, 123
- ? (semnul întrebării), 153
- ?: (operator ternar sau semnul întrebării-două puncte), 94
- @ (simbolul „at”)
 - eliminarea mesajelor de eroare, 105, 335
 - prefixarea apelurilor de funcții, 219
- [] (paranteze drepte)
 - asociere cu tablouri, 62
 - indicarea variabilelor tablou, 123
- \ (caracter slash orientat înapoi)
 - emiterea de comenzi UNIX lungi, 370
 - scrierea la ieșire a numelor variabilelor, 64
 - separarea componentelor unei căi, 177
- \” (escape ghilimele duble), 27-28
- \n (escape salt la linie nouă), 27-28
- \r (escape retur de car), 27-28
- \t (escape caracter de tabulare pe orizontală), 27-28

\\ (pereche de caractere escape slash orientate înapoi)
 secvență escape în PHP, 27-28
 separarea componentelor unei căi, 177
 ^ (caracter accent circumflex), inversarea semnificației expresiilor cu, 153
 _ (caracter de subliniere), 28
 | (bară verticală), 394
 || (bare verticale duble)
 operatorul SAU, 187
 precedența redusă a, 88

A

ACCEPT, atribut, 49
 acces. *vezi* și privilegii
 la coloane specificate dintr-o bază de date MySQL, 251
 obținerea accesului exclusiv la fișier, 195
 privilegii în baza de date MySQL, 250
 accesul la date
 de la variabile de mediu, 67
 de la variabile de mediu, 67
 de la variabile de mediu, proiect, vizualizare, 69
 de la variabile de mediu, test de evaluare, întrebări, 70
 de la variabile de mediu, test de evaluare, răspunsuri, 347
 via formulare HTML, 61
 via formulare HTML, agenda cu adrese de e-mail, 65
 via formulare HTML, expedierea datelor de ieșire către browser, 62
 via formulare HTML, șiruri care înglobează valorile variabilelor, 64
 via formulare HTML, test de evaluare, întrebări, 70
 via formulare HTML, test de evaluare, răspunsuri, 347
 via formulare HTML, vedere de ansamblu, 61
 addslashes(), funcție
 asigurarea datelor introduse de utilizator, 391

conversia (escape) a șirurilor, 291
 conversia (escape) a textelor HTML, 291
Administering Apache (Arnold), 393
 adrese URL
 codificarea caracterelor speciale, 56
 conversie în sens escape, 292
 crearea structurii HTML, 40
 depanare, 22
 execuția scripturilor PHP prin intermediul, 21
 expedierea datelor prin intermediul, 55
 funcții pentru, 420
 afișare
 conținutul fișierelor, 191
 fișierul în totalitate, 188
 variabile șablon, 322-323
 agendă cu adrese, proiect de navigare în creare, 205
 utilizarea bazei de date MySQL, 297
 agregarea datelor dintr-o bază de date, 258
 Allaire HomeSite, program de asistență în editarea textelor, 15
 ALTER TABLE, comandă, 249
 ampersand (&), 114
 ampersand dublu (&&), 88
 amputare, funcții pentru șiruri, 146
 analiză, erori de, 330
 analiză, variabile șablon, 322-323
 AND (SI), operatori, 88
 antete de cerere, definiție, 161
 antete de entitate, 161
 antete de mesaj, 222
 antete generale, definiție, 161
 antete HTML, 161
 antetul funcției, 108
 anunțuri, mesaje de eroare, 330
 Apache, configurare, 393
 Apache, instalare
 pornirea serviciului, 358
 sub Red Hat Linux 7.1, 356
 sub Windows, 361
 testare, 359
 apelare, definiție, 105
 argument domeniu, 165

argumente
 definirea funcțiilor și, 108
 evitarea erorilor la rulare, 331
 funcții definite de utilizator și, 109
 modificarea valorilor și a referințelor, 115
 trecere în revistă, 35, 105
 argumente de tip expirare
 crearea variabilelor cookie, 161
 prestabilite, 110
 specificare, 166
 array(), funcție, 125
 array_search(), funcție, 133
 AS, clauză, 259
 ASCII, coduri
 definiție, 140
 funcții utile pentru, 141
 secvențe escape care folosesc, 140
 tabel cu, 384
 ASCII, fișiere, deschidere în Microsoft Windows, 187
 asigurarea datelor, 237-238
 asigurarea independenței datelor, 237
 asistență, aptitudini HTML, 43
 assign(), metodă, 321
 asterisc (*), 153
 asterisc, caracter slash orientat înainte (*//), 18
 atomice, valori, 244
 atribuire, instrucțiuni
 asocierea valorilor la variabile, 29-30
 atribuirea rapidă a operatorului de adunare (+=), 76
 compararea ecuațiilor matematice cu, 30
 crearea tablourilor, 123
 attribute
 obținerea atributelor fișierelor, 182
 tabele și ~ale bazelor de date, 249
 autentificare. *vezi* și variabile cookie
 a utilizatorilor, 391
 variabile de mediu pentru autentificarea clientului, 69
 autentificarea clientului, 69
 autorizarea utilizatorilor, 391
 avertismente, mesaje de eroare, 330

B

baleierea datelor de ieșire formate, 154
 bară verticală (|), 394
 bare verticale duble (||), 88, 187
 basename(), funcție, 203
 BASH (interfață shell Bourne-again), 369
 bazat pe obiecte, sau orientat spre obiecte, 305
 baze de date relaționale, implementare
 crearea bazelor de date relaționale MySQL, 248
 crearea bazelor de date relaționale MySQL, modificarea tabelor, 249
 crearea bazelor de date relaționale MySQL, ștergerea tabelor, 249
 baze de date relaționale, interogări SQL
 agregare, 258
 expresii și funcții, 260
 grupare, 259
 modificarea datelor, 255
 proiect, crearea și întreținerea, 265
 sortare, 258, 260
 trecere în revistă, 252
 uniri, 263
 baze de date, securitate, 392
 baze de date. *vezi* SGBD (sistem de gestiune a bazelor de date); baze de date MySQL;
 baze de date relaționale
 binare, fișiere, 187
 booleene, expresii, 86 *vezi* și expresii condiționale
 Bourne, interfață shell, 369
 break, instrucțiuni
 încheierea iterației, 128
 scriere, 93
 browsere
 construcția expresiilor șir, 64
 expedierea datelor de ieșire la, 62
 identificarea ~lor client, 69
 buclă, variabile, 95-96
 bucle, instrucțiuni
 scrierea instrucțiunilor while, do while și for, 97
 terminare a, 128

butoare cu imagini
 creare, 53
 proiect, crearea unei pagini HTML cu mai multe formulare, 57
 butoare radio, creare, 47
 buton de expediere, butoane de tip
 creare, 41
 crearea unei pagini cu mai multe formulare, 54
 lucrul cu mai multe formulare, 43
 proiect, crearea unei pagini HTML cu mai multe formulare, 57
 utilizarea unui buton imagine sau, 53
 byreference(), funcție, 115
 byvalue(), funcție, 115

C

calculatoare
 scrierea și execuția datelor, 78
 scrierea și execuția unor ~ simple, 77
 cale de includere, definiție, 187
 cale, argumente
 browsere care impun, 166
 specificarea accesului la variabilele cookie folosind, 164
 caracter accent circumflex (^), inversarea semnificației expresiilor cu, 153
 caracter de salt la linie nouă, secvență escape pentru, 27-28
 caracter de subliniere (_), 28
 caracter de tabulare pe orizontală, secvență escape pentru, 27-28
 caracter slash orientat înainte (/)
 accesul la scripturile PHP prin intermediul adreselor URL, 20
 componentele căilor indicate de, 365
 scrierea instrucțiunilor PHP, 17
 simbolizarea catalogului rădăcină, 177, 365
 caracter slash orientat înainte, asterisc (/ *), 18
 caracter slash orientat înapoi (\)
 emiterea de comenzi UNIX lungi, 370
 scrierea la ieșire a numelor variabilelor, 64
 separarea componentelor unei căi, 177
 caracter, funcții de tip, 398
 caractere de tip spațiu alb, amputare, 146
 caractere slash orientate înapoi (\\)
 secvență escape pentru backslash, 27-28
 separarea componentelor unei căi, 177
 case, scrierea instrucțiunilor ~, 94
 casetă text aglomerată, proiect, 50
 casete cu text
 crearea de ~ personalizate, 45-46
 trecere în revistă, 41
 casete de validare, creare, 47
 cat, operator, 34
 cataloage, 202
 citirea conținutului, 203
 creare, 204
 funcții pentru, 400
 manipularea căilor, 202
 ștergere, 204
 test de evaluare, întrebări, 208
 test de evaluare, răspunsuri, 351
 vizualizarea și modificarea privilegiilor din, 203
 cataloage ascunse, vizualizare, 179
 cataloage curente de lucru
 definiție, 178
 determinare, 178
 modificare, 178, 201, 372
 obținere, 201
 simbol pentru, 367
 cataloage de bază, 178, 367
 cataloage părinte
 definiție, 365
 cataloage, UNIX, 177
 atribuirea permisiunilor, 367
 copiere, 375-376
 creare, 180, 375
 determinarea și modificarea ~lor curente de lucru, 178
 modificarea numelui, 376
 modificarea ~lor curente de lucru, 372
 privilegiu, 180
 ștergere, 180, 375-376
 trecere în revistă, 177, 365
 vizualizarea conținutului, 179, 373
 catalog de bază, 366

căi
 manipulare, 202
 trecere în revistă, 365
 UNIX, 365
 căi absolute
 definiție, 367
 pentru cataloage, 177
 pentru fișiere, 178
 căi relative
 definiție, 178, 366
 simboluri care exprimă, 367
 căutare, funcții de ~ pentru sub-șiruri, 149
 câmpuri ascunse, creare, 48
 cd path, comandă, 179
 chdir(), funcție, 201
 CHECKED, atribut, 47
 chei
 element, 31
 externe, 236, 242
 primare, 234, 242
 chei externe
 identificare, 242
 structura bazelor de date relaționale și, 236
 chei primare compozite
 definiție, 234
 tabele din baze de date care folosesc, 244
 chgrp(), funcție, 185
 chgrp, comandă, UNIX, 173
 chmod(), funcție
 modificarea privilegiilor pentru cataloage, 203
 chmod, comandă, UNIX, 175
 chown(), funcție, 185
 chown, comandă, UNIX, 173
 chr(n), funcție, 141
 clasă de bază, 304
 clase, 302
 definire și instanțiere, 305
 definirea constructorilor, 307
 funcții pentru, 398
 instanțierea obiectelor, 306
 moștenire, 309
 moștenire, invocarea metodelor redefinite, 311
 moștenire, redefinirea metodelor, 310
 orientare spre obiecte, 302
 orientare spre obiecte, bazat pe obiecte sau, 305
 orientare spre obiecte, dezvoltarea PHP și, 308
 orientare spre obiecte, moștenire și, 304
 orientare spre obiecte, vedere de ansamblu, 302
 proiect, lucru cu obiecte, 314
 tablouri cu obiecte, 312
 test de evaluare, întrebări, 317
 test de evaluare, răspunsuri, 354
 clase copil
 definiție, 304
 invocarea metodelor predefinite, 312
 metode redefinite, 310
 utilizarea tablourilor cu obiecte, 312
 clase derivate, definiție, 304
 clase părinte
 definiție, 304
 invocarea unei metode redefinite, 312
 metode redefinite, 310
 utilizarea tablourilor cu obiecte, 312
 class.FastTemplate.php, 321
 closedir(), funcție, 203
 coloane de tabel cu auto-incrementare
 determinarea valorilor funcției
 last_insert_id(), 280
 trecere în revistă, 275-276
 coloane, *vezi* și seturi de rezultate
 identificare și grupare, 241
 specificarea accesului, 251
 structura bazelor de date relaționale și, 234
 utilizarea interogărilor SQL cu, 252
 coloanele specificate, 251
 COLS, atribut, 45-46
 comanda de globalizare a numelor, 376
 comentarii, 17
 comenzi
 emiterea de ~ UNIX, 370
 vizualizarea datelor de ieșire ale ~lor, 374
 comparație între funcții și ~, 306
 comparație între instrucțiunile de atribuire și ecuațiile matematice, 30

concatenare, operator de (.)
 aliasuri pentru cataloage, 178
 trecere în revistă, 34
 unirea a două expresii șir, 62
 conjuncție, operatori de, 88
 consemnare, 335
 consemnarea în jurnal
 a datelor, 336
 a mesajelor de eroare, 335
 funcții pentru lucrul cu coduri ASCII, 400
 constante, 71
 constructori (metode constructor)
 definiție, 304
 trecere în revistă, 307
 continue, instrucțiuni, 129
 controale. *vezi* formulare HTML, controale
 conturi de utilizator
 crearea conturilor nobody, 181
 proprietate și permisiuni, 368
 rădăcină sau super-utilizator, 176
 trecere în revistă, 367
 UNIX, 367
 conversia (în sens escape) a textelor, 291
 conversie automată de tip, 74
 conversie de tip manuală, 75
 conversie forțată, definiție, 75
 conversii forțate de tip, 75
 copierea
 fișierelor, 198
 în sistemul de fișiere UNIX, 172
 mesajelor de e-mail, 231
 copy(), funcție, 198
 corpul funcției
 definirea funcțiilor, 108
 execuție, 109
 count(), funcție, 127
 cp, comandă, 172
 CREATE TABLE, comandă, 248
 curente, cataloage de lucru. *vezi* cataloage de lucru curente
 current(), funcție, 131
 cutii poștale
 accesul la ~ POP, 224
 deschidere, 213
 închidere, 216
 obținerea de informații despre, 217

D

date
 consemnare, 336
 pericole de securitate la adresa ~lor, 389
 date de ieșire formate
 baleiere, 154
 creare, 143
 date de ieșire, scripturi PHP
 evitarea erorilor în logica de program, 332
 expedierea spre browserul Web, 16-17, 62
 funcții de control, 412
 date, în baze de date relaționale
 asigurare, 237-238
 furnizarea independenței, 237
 modificări, 255
 organizare cu, 237-238
 partajare, 236
 DDL (Data Definition Language), 248
 de acces, 250
 modelare E-R, 240
 modelare E-R, gruparea coloanelor în entități, 241
 modelare E-R, identificarea cheilor externe, 242
 modelare E-R, identificarea cheilor primare, 242
 modelare E-R, identificarea coloanelor, 241
 modelare E-R, rafinare, 246
 normalizare, 244
 normalizare, dependență de cheile primare, 245
 normalizare, valori atomice, 244
 proiectare, 240
 dechex(n), 141
 decoct(n), 141
 define(), funcție
 utilizare în constante, 71
 utilizarea șabloanelor, 321
 defined(), funcție, 72
 definire dinamică a tipurilor
 conversie automată, 74
 conversie manuală, 75
 definiție, 74
 DELETE, comandă, 257
 DELETE, interogări, 271

depanare. *vezi* și programe PHP, depanare
 adrese URL, 22
 scripturi PHP, 22-24
 depășirea stivei, 110
 deschiderea fișierelor, 185
 deschiderea sesiunii de lucru
 deschiderea și închiderea sesiunii de lucru, 369, 371-372
 proiect, crearea unei pagini simple de deschidere a sesiunii de lucru, 166
 die(), funcție, 187
 directive (%), 142
 dirname(), funcție, 202
 discuri, utilizarea spațiului, 377
 disjuncție, operatori de, 88
 divide et impera, tehnică, 338
 diviziune, operator de (/), 33
 DML (Data Manipulation Language)
 modificarea datelor din baza de date, 255
 utilizarea interogărilor SQL, 252
 do while, instrucțiuni
 scriere, 97
 utilizare, 98
 domeniu de existență
 comparație între variabilele locale și cele globale, 114
 dosare
 definiție, 177, 365
 manipularea mesajelor de e-mail, comutare, 215
 manipulară mesajelor de e-mail, trecere în revistă, 229
 DROP TABLE, comandă, 249
 duble (numere duble)
 conversie automată de tip, 74
 trecere în revistă, 25
 dump(), metodă, 313
 dump_headers(), funcție, 222
 dump_mailbox_status(), funcție, 218
 durata de viață, variabilă, 113

E

e-mail, 209
 comenzi UNIX pentru citirea și expedierea, 372
 pericol de securitate reprezentat de, 390

recepționare, 213
 recepționare, accesul la cutia poștală POP, 224
 recepționare, antetele mesajului, 222
 recepționare, comutarea între dosare, 215
 recepționare, corpul mesajului, 221
 recepționare, deschiderea cutiei poștale, 213
 recepționare, informații despre cutia poștală, 217
 recepționare, informații despre mesaj, 220
 recepționare, închiderea cutiei poștale, 216
 recepționare, lista mesajelor, 219
 recepționare, lucrul cu identificatori de mesaj, 220
 recepționare, marcarea mesajelor pentru ștergere, 223
 test de evaluare, întrebări, 232
 test de evaluare, răspunsuri, 352
 e-mail, dosare de, 229
 afișarea ~lor existente, 229
 comutare, 215
 copierea mesajelor în, 231
 creare, 230
 funcții IMAP utile, 232
 modificarea numelui, 230
 mutarea mesajelor în, 232
 ștergere, 231
 e-mail, proiecte
 crearea agendei cu adrese, 65
 crearea unui program de navigare, 225
 crearea unui script de trimitere a formularului, 210
 validarea adreselor, 155
 E-R, modelare, 240
 gruparea coloanelor în entități, 241
 identificarea cheilor externe, 242
 identificarea cheilor primare, 242
 identificarea coloanelor, 241
 rafinare, 246
 each(), funcție, 132
 echo, instrucțiuni
 procedee de depanare a variabilelor folosind, 339-340
 proiect, agende cu adrese e-mail, 65

trecere în revistă, 17
 triniterea datelor de ieșire la browser, 62
 utilizarea abrevierii <= pentru, 64
 ecuații matematice, 30
 editare, fișiere UNIX, 171, 375
 editoare de texte, 14, 15
 elemente, definiție, 31
 else, instrucțiune, 92
 elseif, instrucțiune, 92
 ENCTYPE, atribut, 49
 ENT_COMPAT, 292
 ENT_NOQUOTES, 292
 ENT_QUOTES, 292
 ereg(), funcție, 153-154, 155
 erori de domeniu
 evitarea mesajelor de eroare, 333
 mesaje de eroare la rulare ca, 331
 erori fatale
 care determină încheierea execuției
 scriptului, 334
 mesaje de eroare la rulare, 330
 error_log(), funcție
 consemnarea datelor arbitrare, 336
 consemnarea mesajelor de eroare, 336
 error_reporting(), funcție, 335, 342
 etichete HTML, delimitarea între caracterele
 <<, 17
 exit(), funcție, 109
 experiența programatorului, 237-238
 expresie de inițializare, 95-96
 expresii condiționale, 86
 metode de formare, 87
 ordinea de precedență, 87
 reguli de definire, 87
 expresii. *vezi și* expresii condiționale
 accesul la datele din bazele de date SQL,
 260
 definiție, 33
 formarea de ~ logice, 254
 Extensible Markup Language (XML), 295,
 297

F

false, valori de tip ~
 definire și utilizare, 86
 scrierea instrucțiunilor if complexe, 91
 FastPrint, metodă, 322-323
 FastTemplate, clasă
 dezvoltare, 319
 obținerea mesajelor de eroare, 322
 FastTemplate, instanțierea obiectelor \$, 320
 fclose(), funcție, 188
 feof(), funcție, 190
 fgetc(), funcție, 188
 fgets(), funcție, 189
 filegroup(), funcție, 182
 fileowner(), funcție, 182
 fileperms(), funcție, 203
 filesize(), funcție, 188
 find, comandă, 379
 fișiere, 181
 acces exclusiv la, 195
 blocare, 197
 căi ale ~lor, 367
 citire, 188
 citire, a unui ~ întreg linie cu linie, 190
 citire, afișarea conținutului, 191
 copiere, 198
 definiție, 169
 deschidere, 185
 închidere, 188
 modificarea numelui, 199
 navigare, 191
 obținerea atributelor, 182
 posesor, modificare, 185
 privilegii, modificare, 184
 privilegii, probleme de, 181
 proiecte, contor al accesului la o pagină
 Web, 194
 proiecte, contor îmbunătățit al accesului la
 o pagină Web, 197
 proiecte, program de navigare în agenda cu
 adrese, 205
 proprietate, probleme de, 181
 scriere, 193
 ștergere, 199
 test de evaluare, întrebări, 208

test de evaluare, răspunsuri, 351
 utilizarea ~lor incluse, 106
 verificarea finalizării operațiilor cu fișiere,
 187
 fișiere ascunse, vizualizare, 179
 fișiere atașate, poșta electronică, 211
 fișiere, UNIX, 169
 compararea textelor, 380
 comprimate, 380
 copiere, 172, 375-376
 denumire, 169, 365
 determinarea tipului de ~, 380
 editare, 171, 375
 găsire, 379
 modificarea numelui, 173, 376
 permisiuni, 367, 378
 privilegii, 174
 proprietate, 173, 378
 ștergere, 172, 375
 tipuri, 173
 transferul de ~ spre și dinspre gazde aflate
 la distanță, 382
 trecere în revistă, 169, 365
 vizualizare, 171
 vizualizarea conținutului unui catalog, 373
 vizualizarea conținutului ~lor și a datelor
 de ieșire ale comenzilor, 374
 flock(), funcție, 196
 fopen(), funcție
 deschiderea fișierelor, 185
 verificarea finalizării operațiilor cu fișiere,
 187
 foreach, instrucțiuni
 parcurea iterativă a tablourilor
 non-secvențiale, 129
 prelucrarea rezultatelor interogărilor
 SELECT, 279
 FORM, etichetă
 crearea structurii HTML, 40
 expedierea conținutului fișierului către
 servere, 48
 încorporarea controalelor, 41
 lucrul cu mai multe formulare, 42
 formate externe, 25
 formate interne, 25-27
 formulare de încărcare, creare, 200
 formulare HTML, accesul la date, 61
 construcția șirurilor, 64
 expedierea de date de ieșire spre browser,
 62
 proiect, agende cu adrese de e-mail, 65
 test de evaluare, întrebări, 70
 test de evaluare, răspunsuri, 347
 trecere în revistă, 61
 formulare HTML, controale, 45
 butoane radio, 47
 casete cu parolă, 46
 casete cu text personalizate, 45-46
 casete de validare, 47
 câmpuri ascunse, 48
 expedierea conținutului fișierelor către
 servere, 49
 încorporare, 41
 îndrumări pentru proiectare, 38-39
 proiect, casetă aglomerată, 50
 selecții, 47
 suprafețe cu text, 46
 formulare HTML, creare, 38-60
 încărcarea formularelor, 200
 încorporarea controalelor, 41
 lucrul cu mai multe formulare, 42
 noțiuni elementare de proiectare, 38-39
 proiect, vizualizarea câmpurilor din
 formulare, 44
 reactualizarea cunoștințelor despre, 43
 structura HTML, 40
 test de evaluare, întrebări, 60
 test de evaluare, răspunsuri, 346
 formulare HTML, expediere, 52
 crearea butonului de reinițializare, 54
 crearea paginilor cu mai multe formulare,
 54
 expedierea datelor prin intermediul adresei
 URL a scripturilor, 55
 pagină cu mai multe formulare, 57
 test de evaluare, întrebări, 60
 utilizarea unei imagini pentru expedierea
 datelor, 52
 fpassthru(), funcție, 191
 fread(), funcție, 188

fseek(), funcție, 192
 FTP (File Transfer Protocol)
 funcții, 403
 încărcarea scripturilor FTP prin intermediul, 19-21
 ftp, comandă, 382
 funcție de tratare a erorilor, 334
 funcții
 acceptate numai de MySQL, 262
 accesul la funcțiile de date SQL, 260
 compararea metodelor cu ~le, 306
 definiție, 108
 definiție, apelarea ~lor definite de utilizator, 109
 definiție, argumente prestabilite, 110
 definiție, terminarea execuției, 109
 definiție, trecere în revistă, 108
 definiție, ~ recursivă, 110
 diverse, 409
 executarea operațiilor comune, 35
 execuție, 105
 proiect, crearea formularului pentru persoane de contact, 116
 test de evaluare, întrebări, 121
 test de evaluare, răspunsuri, 349
 tipuri, 394
 traversarea tablourilor, 131
 URL, 420
 utilizarea fișierelor incluse, 106
 utilizarea referințelor, 115
 utilizarea variabilelor globale, 112
 utilizarea variabilelor locale și a celor statice, 113
 funcții ale sistemului de fișiere, 400
 funcții calendar, 397
 funcții criptografice, 392
 funcții de comparație, șiruri, 148
 funcții de date
 MySQL, 262
 PHP, 399
 funcții de execuție a programelor, 413
 funcții de extragere, 149
 funcții de înlocuire, sub-șiruri, 151
 funcții de verificare a ortografiei, 397
 funcții definite de utilizator, apelare, 109

funcții matematice
 MySQL, 261
 PHP, 407
 funcții pentru mesaje poștale, 407
 funcții, PHP, 394
 ale sistemului de fișiere, 400
 calendar, 397
 clasă/obiect, 398
 dată și oră, 399
 de control al ieșirii, 412
 de execuție a programelor, 413
 de manipulare a sesiunilor, 416
 de tip caracter, 398
 de tip tablou, 395
 de variabilă, 420
 expresii regulate extinse POSIX, 415
 FTP, 403
 HTTP, 404
 IMAP, POP3 și NNTP, 404
 matematice, 407
 MySQL, 410
 opțiuni și informații, 412
 pentru manipularea cataloagelor, 400
 pentru poștă, 407
 pentru tratarea și consemnarea erorilor, 400
 pentru verificarea ortografiei, 397
 POSIX, 414
 prototipuri, 394
 șir, 416
 fwrite(), funcție, 193

G

gazde
 comandă UNIX pentru verificarea ~lor, 371-372
 deschiderea sesiunii de lucru cu un sistem UNIX, 369
 transferarea fișierelor spre și dinspre ~ aflate la distanță, 382
 gestiunea bazelor de date. *vezi* baze de date
 MySQL, instrumente de gestiune a datelor
 GET, metodă, 40
 getcwd(), funcție, 201
 gettype(), funcție, 76

get_lock(), funcție, 262
 get_object_vars(), funcție, 218
 ghilimele duble ("...")
 delimitarea șirurilor între, 28, 141
 secvență PHP escape pentru, 27-28
 ghilimele magice
 alegerea opțiunilor pentru activare, 292
 trecere în revistă, 289
 GNU zip, metodă
 comenzi pentru, 380
 lucrul cu fișiere tar, 381
 GRANT, comandă, 250
 grep, comandă, 379
 GROUP BY, clauză
 gruparea bazei de date cu, 259
 sortarea în funcție de valori calculate sau de valorile din coloane, 259-260
 grupuri
 accesul la datele din bazele de date SQL, 259
 definirea proprietății asupra unui fișier, 173
 tabelele din baze de date, 244
 grupuri de informare, resurse PHP pentru, 362-363
 grupuri de instrucțiuni
 definiție, 91
 scriere, 95-96
 grupuri de utilizatori
 definiție, 367
 proprietate asupra unui fișier, 173
 UNIX, 367

H

hackeri, 389
 HAVING, clauză
 gruparea bazei de date, 259-260
 sortarea pe valori calculate sau pe valori ale coloanelor, 259-260
 hexdec(n), 141
 hibe. *vezi* programe PHP, depanare
 HTML: *A Beginner's Guide* (Willard), 43
 htmlentities(), funcție, 292
 htmlspecialchars(), 291
 HTTP, funcții, 404

HTTP_POST_VARS ["data"], 62
 HTTP_USER_AGENT, variabilă de mediu, 69

I, K

IDE (medii de dezvoltare integrată), 15
 identificatori de legătură, 271
 identificatori de mesaj, 220
 if, instrucțiuni
 complexe, 91
 proiect, testarea valorilor numerice, 89
 simple, 89
 IIS, instalare sub Windows NT, 361
 IMAP, 214
 IMAP (Interim Mail Access Protocol), 213
 accesul la cutia poștală POP, 224
 funcții, 232, 404
 închiderea cutiei poștale, 216
 obținerea de informații despre cutia poștală, 217
 obținerea de informații despre mesaj, 220
 imap_close(), funcție, 216
 imap_errors(), funcție, 215
 imap_fetch_overview(), funcție, 220
 imap_headerinfo(), funcție, 222
 imap_headers(), funcție, 219
 imap_mailboxmsginfo(), funcție, 218
 imap_num_msg(), funcție, 217
 imap_open(), funcție
 comutarea dosarelor cu mesaje, 216
 deschiderea cutiilor poștale, 213
 imap_reopen(), funcție, 216
 Implementing Directory Services (Reed), 295
 index, definiție, 95-96
 INSERT, comandă
 adăugarea mai multor rânduri dintr-un tabel în baza de date, 257
 modificarea datelor dintr-o bază de date, 255
 INSERT, interogări, 271
 instalarea PHP, 356
 versiuni sub UNIX și Linux, 359
 Windows NT/2000 și 95/98, 360

instalarea PHP, Red Hat Linux 7.1, 356
 configurarea MySQL, 358
 configurarea serverului Apache, 356
 instalarea IMAP, 357
 instalarea MySQL, 357
 pornirea serverului Apache, 358
 testarea instalării, 359
 trecere în revistă, 357

instanțiere
 clase, 305
 obiecte, 306
 obiecte FastTemplate, 320

instrucțiuni condiționale, 85-104
 for, instrucțiuni, 95-96
 if, instrucțiuni complexe, 91
 if, instrucțiuni simple, 89
 instrucțiunile switch, default și break, 93
 instrucțiunile while și do while, 97
 proiect, validarea datelor introduse de utilizator, 99
 test de evaluare, întrebări, 104
 test de evaluare, răspunsuri, 348
 valori adevărat/fals și, 86

instrucțiuni if imbricate, 92

instrucțiuni include
 instanțierea obiectelor FastTemplate, 320
 proiect, crearea formularului cu persoane de contact, 116
 protecția numelor de utilizator și a parolelor, 393
 trecere în revistă, 106

instrucțiuni, 7-8. *vezi și* instrucțiuni condiționale

instrumentarea programelor, definiție, 339-340

instrumente
 facilități PHP, 296
 scrierea programelor PHP, 14

Interim Mail Access Protocol. *vezi* IMAP (Interim Mail Access Protocol)

interogare, șiruri de, 56

interogări
 definiție, 271
 structura unei baze de date relaționale și, 237

invocarea unei metode redefinite, 311

încapsulare
 definiție, 302
 încălcare, 308

încărcarea scripturilor PHP, 19-21

înmulțire, operator (*), 33

întregi (numere întregi)
 conversie automată de tip, 74
 scrierea instrucțiunilor switch, 94
 trecere în revistă, 25

key(), funcție, 132

L

LDAP (Lightweight Directory Access Protocol), 295

libmcrypt, bibliotecă, 165

Lightweight Directory Access Protocol. *vezi* LDAP (Lightweight Directory Access Protocol)

limbaje compilate, 329

limbaje cu tipuri bine definite, definiție, 331

limbaje cu tipuri slab definite, definiție, 331

linie, numere de, 329

linii de program
 evitarea erorilor în logica de program, 332
 evitarea erorilor sintactice, 330
 înțelegerea hibelor de program, 328
 procedeul de depanare divide et impera, 338
 scrierea de ~ schelet, 14-15

Linux. *vezi și* Red Hat Linux 7.1

editoare de text pentru scrierea de scripturi PHP, 14

instalarea ODBC, 294

instalarea PHP, 359

instalarea suportului LDAP, 295

încărcarea scripturilor PHP prin intermediul ~, 19-21

suport PostgreSQL pentru, 294

list(), funcție, 133

liste de corespondență, resurse PHP pentru, 362-363

list_messages(), funcție, 220

literale
 proiect, vizualizarea valorilor variabilelor PHP, 30
 vedere de ansamblu, 28-30

locale, variabile, 113

locate, comandă, 379

logici, operatori
 formarea expresiilor condiționale complexe, 87
 formarea expresiilor logice, 254
 precedență și, 88
 tabel cu ~ din MySQL, 260-261

ls, comandă
 datele de ieșire ale, 171
 definirea privilegiilor asupra unui fișier, 174
 definirea proprietății asupra unui fișier, 173
 privilegiu speciale folosite cu, 177
 vizualizarea conținutului unui catalog, 179, 373

lucrul cu obiecte, 314

lucrul cu obiectul FastTemplate, 322

lucrul cu șabloane, 323

lungime
 a coloanelor din setul de rezultate, 281
 manipularea ~ii șirului, 146
 lungime, argumente de tip, 189

M

mail(), funcție, 209

mail, comandă, 372

manual, pagini de, 153-154, 377

master, tabele, 263

master-detaliu, relații de tip, 263

MAXLENGTH, atribut, 45-46

Mcrypt, funcții, 165

mecanisme de tratare a excepțiilor, 333

medii de dezvoltare integrată (IDE), 15

mesaj, antete de
 definiție, 209
 obținere, 222

mesaj, corpul ~ului
 expediere, 209
 obținere, 221

mesaje
 marcarea mesajelor IMAP în vederea ștergerii, 223
 mutare, 232
 obținerea de informații despre, 220
 obținerea unei liste cu, 219

mesaje de eroare, 333

mesaje de eroare la rulare, 421-423. *vezi și* mesaje de rulare

evitare, 333
 funcții pentru, 400

metacaractere
 comandă UNIX pentru, 376
 neîncrederea în datele introduse de utilizator din motive de ~, 391

METHOD, atribut, 40, 50

metode
 caracteristici ale ~lor, 303
 invocare, 311
 trecere în revistă, 310

metode accesori, 307

metode de obținere, 307

metode redefinite

Microsoft Windows
 comenzi pentru fișiere și cataloage, 169
 deschiderea fișierelor binare și a fișierelor ASCII, 187
 deschiderea sesiunii de lucru cu un sistem UNIX, 369
 editoare de texte pentru scripturi PHP, 14
 instalarea PHP pe Windows 95/98, 359
 instalarea PHP pe Windows NT/2000, 360
 instalarea PWS sub, 361
 încărcarea scripturilor PHP prin intermediul, 19-21
 separarea componentelor unei căi, 177
 suport ODBC pentru, 294
 utilizarea programului Notepad pentru a scrie scripturi PHP, 14

MIME (Multipurpose Internet Mail Extensions), tip, 48

mkdir(), funcție, 204

mkdir, comandă, UNIX, 180

mktime(), funcție
 specificarea momentului expirării unei variabile cookie, 161

modelarea entitate-relație în bazele de date relaționale, 241

modificarea privilegiilor pentru fișiere, 184

modulo, operator (%), 34

more, comandă
 vizualizarea conținutului fișierelor și a datelor de ieșire ale comenzilor, 374
 vizualizarea fișierelor, 171

moștenire, 309
 invocarea unei metode redefinite, 311
 redefinirea metodelor, 310
 trecere în revistă, 304
 utilizare, 309

motive logice pentru, 236

Multiple Internet Mail Extensions (MIME), tip, 48

MULTIPLE, atribut, 47

mutator, metode, 307

MySQL, accesul la bazele de date ~, 269
 comparație între Postgresql și, 239
 conectare la servere, 270
 crearea programului de navigare prin agenda cu adrese, 297
 închiderea conexiunii cu serverul, 273
 selectarea bazelor de date, 271
 test de evaluare, întrebări, 301
 test de evaluare, răspunsuri, 353
 trecere în revistă, 270
 verificarea și suprimarea erorilor, 271-272

MySQL, baze de date
 configurare sub Red Hat Linux 7.1, 358
 funcții, 410
 indicatoare, 282
 instalare sub Red Hat Linux 7.1, 357
 tabele, 283
 tipuri, 282

MySQL, crearea bazelor de date ~, 309-314.
vezi și seturi de rezultate
 acordarea accesului la coloanele specificate, 251
 acordarea și revocarea privilegiilor de acces, 250
 modificarea tabelor, 249
 principalele tipuri de date, 246
 ștergerea tabelor, 249
 trecere în revistă, 248

MySQL, ghilimele în bazele de date, 289
 conversia în sens escape a adreselor URL, 292
 conversia în sens escape a textelor HTML, 291
 conversia în sens escape și anularea conversiei, 291
 ghilimele magice, 289, 292

MySQL, instrumente de gestiune a datelor din bazele de date, 293
 instrumente și facilități PHP, 296
 LDAP, 295
 ODBC, 294
 Postgresql, 293
 XML, 295, 297

MySQL, interogări SQL pentru bazele de date ~, 271
 coloane de tabel cu auto-incrementare, 275-276
 interogări care nu returnează rânduri de tabel, 271
 mysql_query(), funcție, 274
 prelucrarea rezultatelor interogării SELECT, 277

MySQL, server
 conectare la, 270
 determinarea bazelor de date găzduite de, 286

mysql_affected_rows(), funcție
 determinarea reușitei, 276
 verificarea interogărilor, 271

MYSQL_ASSOC, 279

mysql_close(), funcție, 273

mysql_connect(), funcție, 270, 287

mysql_data_seek(), 284

mysql_dbname(), funcție, 287

mysql_error(), funcție, 271-272

mysql_fetch_array(), funcție
 accesul nesecvențial la coloanele din setul de rezultate, 284
 prelucrarea rezultatelor interogărilor SELECT, 279

mysql_fetch_field(), funcție, 283, 288

mysql_fetch_row(), funcție
 accesul nesecvențial la coloanele din setul de rezultate, 284
 prelucrarea rezultatelor interogărilor SELECT, 278

mysql_field_flags(), funcție, 282

mysql_field_len(), funcție, 281

mysql_field_name(), funcție, 281

mysql_field_table(), 282

mysql_field_type(), funcție, 282

mysql_insert_id(), funcție
 determinarea unei valori viabile a funcției, 280
 la returnarea unui rezultat incorect, 277
 utilizarea coloanelor cu auto-incrementare, 276

mysql_list_dbs(), funcție, 286

mysql_list_fields(), funcție, 288

mysql_list_fields(), funcție, 288

mysql_num_fields(), funcție, 280

mysql_num_rows(), funcție, 287

mysql_num_rows(), funcție
 determinarea bazelor de date găzduite de serverul MySQL, 285
 prelucrarea rezultatelor interogărilor SELECT, 277
 recepționarea mesajului de eșec, 287

mysql_query(), funcție, 274, 277

mysql_select_db(), funcție, 271, 272

mysql_tablename(), funcție, 286

N

n12br(), funcție, 292

NAME, atribut
 crearea butoanelor radio, 47
 crearea câmpurilor ascunse, 48
 crearea selecțiilor, 47
 crearea suprafețelor de text, 46
 verificarea casetelor de validare, 47

natcasesort(), funcție, 135

navigarea în fișiere
 proiect, program de navigare în agenda cu adrese, 205
 trecere în revistă, 191

Network File System (NFS), 21

Network Security: A Beginner's Guide (Maiwald), 393

next(), funcție, 131

NFS (Network File System), 21

NNTP, funcții, 404

nobody, conturi de tip, 181

normalizare, baze de date relaționale, 244
 dependență de cheile primare, 245
 valori atomice, 244

Notepad, scrierea scripturilor PHP cu, 14

NULL, valori, 109, 253

număr hexazecimal (\xnn), 140

number_format(), funcție, 145

nume
 butoane radio și, 47
 casete cu text și, 41, 45-46
 coloane din setul de rezultate, 281
 dosare IMAP, 216, 230
 instrucțiuni require și include, 107
 limbajul SQL folosit cu bazele de date relaționale, 241
 reguli pentru fișierul UNIX, 365
 reguli pentru scripturi PHP, 15
 selecții, 47
 variabile, 28

nume complet determinate, 263

nume, argumente de tip, 161

numele gazdei, 270

numere
 baze diferite pentru, 140
 crearea datelor de ieșire formate pentru, 143
 scriere, 25
 ~le coloanelor din setul de rezultate, 280

numere cu virgulă mobilă
 definiție, 25
 scrierea instrucțiunilor switch, 94

O

obiect, tablouri, 312

obiecte, 302
 definirea funcțiilor constructor, 307
 instanțierea obiectelor, 306
 moștenire, 310
 moștenire, invocarea metodelor redefinite, 311
 moștenire, redefinirea metodelor, 310
 orientare spre obiecte, 302
 orientare spre obiecte, bazat pe obiecte sau, 305

orientare spre obiecte, dezvoltarea PHP și, 308
 orientare spre obiecte, moștenire și, 304
 proiect, lucru cu obiecte, 314
 tablouri, 312
 tablouri cu obiecte, 312
 test de evaluare, întrebări, 317
 test de evaluare, răspunsuri, 354
 obținerea listei mesajelor, 219
 comutarea între dosare, 215
 deschiderea cutiei poștale, 213
 instalarea sub Red Hat Linux 7.1, 357
 lucrul cu identificatori de mesaj, 220
 marcarea mesajelor pentru ștergere, 223
 obținerea antetelor mesajului, 222
 obținerea corpului mesajului, 221
 proiect, crearea unui program de navigare prin mesaje de e-mail, 225
 trecere în revistă, 213
 octal, cifre în, 175
 octal, număr în (xxx), 140
 octdec(n), 141
 octeți, 170
 Open Database Connectivity (ODBC), 294
 opendir(), funcție, 203
 open_mailbox(), funcție, 215
 operanzi
 definiție, 33
 plasarea operatorilor de incrementare/decrementare după, 34
 verificarea ~lor pentru evitarea erorilor la rulare, 331
 operator de adunare (+)
 utilizare aritmetică, 33
 valoare abreviată pentru repetiție, 153
 operator de decrementare (<<-), 34
 operator de incrementare (++)
 plasare după operand, 34
 trecere în revistă, 34
 operatori
 combinarea valorilor din expresii cu, 33
 de comparație folosiți în SQL, 254
 tabel cu ~ PHP, 387
 tabel cu ~ SQL logici, 260-261
 operatori de comparație, SQL, 254
 operatori de unire, 34
 operatori matematici, MySQL, 260-261
 operație, argument, 176
 OPTION, etichete, 47
 OR (SAU), operator
 precedența redusă a, 88
 verificarea finalizării operațiilor cu fișiere, 188
 ord(c), coduri ASCII, 141
 ORDER BY, clauză
 câmp de sortare din baza de date, 258
 sortarea pe valori calculate sau pe valori din coloane, 259-260
 ordinare, caractere, 142

P, Q

pagini cu mai multe formulare
 creare, 54
 expediere, 57
 paranteze (), 64
 paranteze acolade ({})
 identificarea variabilelor de șablon, 320
 specificarea repetițiilor pentru expresiile regulate, 153
 paranteze drepte ([])
 asociere cu tablouri, 62
 indicarea variabilelor tablou, 123
 parole
 comandă UNIX pentru modificarea, 371
 conectare la serverul MySQL, 270
 crearea de casete pentru, 46
 parse(), metodă, 322-323
 partajarea datelor, 236
 partiționare prin echivalență, 341
 pas, expresie, 95-96
 PASSWORD, atribut, 46
 pathinfo(), funcție, 203
 pereche de caractere slash orientate înainte (/ /), 17
 permisiuni, UNIX
 atribuire, 368
 trecere în revistă, 368
 vizualizarea informațiilor despre fișiere, 170

persoane de contact, proiect de formular, 116
 PHP, crearea programelor ~, 13-24
 date de ieșire pentru browserul Web, 16-17
 depanare, 22-24
 documentare, 17
 proiect, 22
 proiect, validarea datelor din formular, 99
 scriere, 14-15
 test de evaluare, întrebări, 24
 test de evaluare, răspunsuri, 345
 PHP, depanarea programelor ~, 417-438. *vezi și* mesaje de eroare
 date de ieșire incorecte sau care lipsesc, 332
 erori de sintaxă, 329
 înțelegerea hibelor, 328
 mesaje de eroare la rulare, 330
 practici, 337
 practici, înțelegere, 339-340
 practici, remediere, 339-340
 practici, reproducerea simptomelor, 337
 practici, stabilirea cu precizie a hibe, 338
 practici, testarea programelor, 341
 practici, trecere în revistă, 337
 proiect, mesaje de eroare PHP, 342
 test de evaluare, întrebări, 344
 test de evaluare, răspunsuri, 355
 trecere în revistă, 327
 PHP, elemente constructive, 25-37
 funcții, 35
 literale și variabile, 28-30
 numere, 25
 operatori, 33
 proiect, calcul în PHP, 36
 proiect, literale și variabile, 30
 scalari și tablouri, 31-32
 șiruri, 27-28
 test de evaluare, întrebări, 37
 test de evaluare, răspunsuri, 346
 PHP, execuția programelor ~, 19-24
 încărcare, 19-21
 proiect, 22
 trecere în revistă, 21
 PHP, instalare. *vezi* instalare PHP
 PHP, resurse, 362
 grupuri de informare, 362-363
 liste de corespondență, 362-363
 situri Web, 362
 phpinfo(), funcție
 testarea instalării PHP, 359
 vizualizarea variabilelor de mediu PHP, 67
 pi(), funcție, 72
 Pico, editor
 comenzi pentru, 172
 editarea fișierelor UNIX, 375, 171
 pine, program client de e-mail, 372
 POP, cutii poștale, 224
 POP3, funcții, 404
 porturi, 271
 POSIX
 funcții, 414
 funcții pentru expresii regulate extinse, 415
 înțelegerea expresiilor regulate, 153-154
 POST, valoare, 40
 PostgreSQL
 trecere în revistă, 293
 utilizare pentru baze de date, 239
 precedență
 operatori PHP, 88
 trecere în revistă, 34
 precizie, specificatori de, 143
 prefixarea apelurilor de funcții, 219
 eliminarea mesajelor de eroare, 105, 335
 prev(), funcție, 131
 primare, chei
 identificare, 242
 normalizarea unei baze de date, 245
 printf(), funcție
 crearea datelor de ieșire formate, 142
 inserția de simboluri ale procentului, 144
 print_body_by_id(), funcție, 222
 print_body_by_num(), funcție, 222
 print_error_stack(), funcție
 deschiderea cutiilor poștale, 215
 închiderea cutiilor poștale, 216
 print_headers(), funcție, 222
 print_mailbox_status(), funcție, 218
 print_overview(), funcție, 221
 print_r(), funcție, 218, 222

privilegii
 bază de date MySQL, acordarea accesului la coloanele specificate, 251
 bază de date MySQL, acordarea și revocarea accesului, 250
 catalog, acces la, 180
 catalog, UNIX, 180
 catalog, vizualizare și modificare, 203
 fișier, modificare, 184
 fișier, probleme legate de, 181
 fișier, trecere în revistă, 174
 privilegii, argument, 176
 programatori
 baze de date relaționale, 237-238
 întreținerea programelor, 305
 mesaje de eroare destinate ~lor, 331
 remedierea hibelor, 339-340
 testarea programelor, 340
 programe corecte din punct de vedere funcțional, 328
 programe de analiză, definiție, 295
 programe driver
 SGBD, 236
 tehnica de depanare divide et impera, 339
 proiect de calcul, script pentru calculul ariei unui cerc, 36
 proiect de generare a știrilor de senzație, 78
 proiect de numărare a deschiderilor unei pagini
 crearea fișierelor și a cataloagelor, 194
 crearea unui ~ îmbunătățit, 197
 proiect, mesaje de eroare PHP, 342
 proiecte
 agendă cu adrese de e-mail, 65
 bază de date MySQL, 265
 calculator simplu, 77
 calculator, date, 78
 calcule în PHP, 36
 casetă HTML aglomerată, 50
 contor al numărului de deschideri al unei pagini, 194
 contor îmbunătățit al numărului de deschideri al unei pagini, 197
 demonstrarea instrucțiunilor condiționale, 99
 formulare pentru persoane de contact, 116, 136

generator de știri de senzație, 78
 mesaje de eroare PHP, 342
 pagină de deschidere a sesiunii de lucru, 166
 primul script PHP, 22
 program de navigare în agenda cu adrese, 205, 297
 program de navigare prin mesajele de e-mail, 225
 program de stabilire a echivalențelor cu expresii regulate, 155
 script de expediere a formularului, 210
 vizualizarea câmpurilor dintr-un formular, 44
 vizualizarea valorilor variabilelor PHP, 30
 vizualizarea variabilelor de mediu, 69
 proprietate
 asupra fișierelor, 181, 185
 fișiere și cataloage UNIX, 368
 fișiere UNIX, 173
 proprietate de grup, 368
 proprietăți, 303
 punct și virgulă (;), 29, 62
 puncte de suspensie (...)
 prototipuri de funcții, 394
 punct_zecimal, argument, 145
 pwd, comandă, 372
 PWS, instalare sub Windows, 361
 quotemeta(), funcție, 291

R

r, privilegiu
 privilegii de catalog, 180
 privilegii de fișier, 175
 racorduri
 crearea formularului pentru persoane de contact, 120
 tehnica divide et impera pentru remedierea hibelor, 339
 rafinare, modele E-R, 246
 rawurlencode(), funcție, 292
 rădăcină, catalog
 alias pentru, 178
 definiție, 365

rădăcină, conturi
 definiție, 176
 modificarea proprietății, 185
 rînduri
 adăugarea mai multor ~ dintr-un tabel la o bază de date, 257
 structura unei baze de date relaționale și, 234
 utilizarea interogărilor SQL, 253
 readdir(), funcție, 203
 readfile(), funcție, 191
 recursive, funcții, 110
 Red Hat Linux 7.1, 357
 configurarea MySQL, 358
 instalarea Apache, 356
 instalarea IMAP, 357
 instalarea MySQL, 357
 instalarea PHP, 357
 instalarea suportului pentru LDAP, 295
 pornirea serviciului Apache, 358
 suportul Postgresql al sistemului ~, 294
 suportul XML al sistemului ~, 297
 testarea instalării, 359
 redefinite, invocare, 311
 redefinite, trecere în revistă, 310
 referințe, 115
 referințe indirecte, 73
 refuzul execuției serviciului, 390
 regresie, teste prin, 340
 regulate, expresii
 caractere echivalente, 152
 formare, 152
 formarea expresiilor logice cu, 254
 proiect, program de stabilire a echivalențelor cu expresii regulate, 155
 specificarea repetițiilor, 153
 reinițializare, crearea butoanelor de ~, 54
 relaționale, baze de date, 234
 decizia privind utilizarea, 239
 eficiență în prelucrarea datelor, 238
 interogare ad-hoc, 237
 organizarea datelor, 237-238
 selectarea tipului de, 239
 test de evaluare, întrebări, 268
 test de evaluare, răspunsuri, 353

relaționali, operatori
 formarea expresiilor condiționale, 86
 tabel cu, 86
 release_lock(), funcție, 262
 REMOTE_ADDR, variabilă de mediu, 69
 rename(), funcție
 modificarea numelui cataloagelor, 204
 modificarea numelui fișierelor, 199
 repetiție, expresii regulate, 153
 require, instrucțiuni
 acces la programe scrise anterior, 106
 protecția numelor de utilizator și a parolelor, 393
 require_once, instrucțiuni, 108
 resource, tipuri, 394
 resurse, PHP, 362
 grupuri de informare, 362-363
 liste de corespondență, 362-363
 securitatea rețelor, 393
 situri Web, 362
 retur de car, secvență escape pentru, 27-28
 return, instrucțiuni
 definirea funcțiilor, 109
 încheierea execuției funcțiilor, 109
 REVOKE, comandă, 250
 rewind(), funcție, 191
 rmrf path, comandă, 180
 rmdir(), funcție, 204
 rmdir, comandă, UNIX, 180
 ROWS, attribute, 45-46
 row_number, 284
 rwx, secvență, 175

S

scalare, valori, 71
 conversie automată de tip, 74
 conversie manuală de tip, 75
 definirea și utilizarea constantelor, 71
 proiect, calculator de date calendaristice, 78
 proiect, calculator simplu, 77
 proiect, generator de știri de senzație, 78
 test de evaluare, întrebări, 83
 test de evaluare, răspunsuri, 348
 trecere în revistă, 31-32
 utilizarea variabilelor dinamice, 72

scp, program, 21
 script de trimitere a datelor într-un formular, creare, 210
 scripturi
 date de ieșire, evitarea erorilor în logica programului, 332
 date de ieșire, expedierea spre un browser Web, 16-17, 62
 date de ieșire, funcții de control, 412
 depanare, 355
 editoare de text pentru scrierea, 14
 expedierea datelor prin intermediul adresei URL a șlor, 55
 limbaje pentru, 329
 proiecte, primul script PHP, 22
 proiecte, script de expediere a formularului, 210
 reguli de denumire, 15
 utilizarea FTP pentru încărcare, 19-21
 „script kiddies“, 389
 secure shell service (SSH)
 deschiderea sesiunii de lucru cu un sistem UNIX, 369
 încărcarea scripturilor PHP, 21
 securitate, 389
 contramăsuri, 391
 date în baze de date relaționale, 237-238
 pericole și riscuri, 389
 secvență de colajonare, definiție, 134
 secvențe escape
 conversia ghilimelelor duble (\"), 27-28
 crearea șirurilor, 139
 scrierea și execuția, 78
 tabel cu ~ folosite în PHP, 139, 383
 secvențiale, tablouri
 căutare prin, 127
 parcursarea prin iterație a, 127
 SELECT, etichetă, 47
 SELECT, interogări
 determinarea structurii, 285
 prelucrarea rezultatelor, 277
 rapoarte referitoare la coloanele din bazele de date, 252
 selecții, creare, 47

semnul egal (=)
 abreviere pentru echo, 64
 asocierea cheilor cu valorile (=>), 123
 instrucțiuni de atribuire, 29
 semnul întrebării (?), 153
 semnul întrebării-două puncte, operator (?:), 94
 sensibilitatea la diferența dintre majuscule și minuscule
 executarea de căutări pentru tablouri, 135
 nume de fișiere UNIX, 169
 nume de variabile PHP, 29
 SQL și, 241
 șiruri, 147
 separator_mii, argument, 145
 serialize(), funcție, 163
 server, antete de răspuns al ~ului, 161
 servere
 adecvat configurate din punct de vedere al securității, 392
 creare, 49
 server MySQL, conectare la, 270
 server MySQL, determinarea bazelor de date găzduite de, 286
 sesiuni, funcții de manipulare a ~lor, 416
 setcookie(), funcție
 crearea fișierelor cookie, 161
 formatul complet al funcției, 164
 settype(), funcție, 76
 seturi de rezultate, 277
 accesul în mod ne-secvențial, 284
 definiție, 277
 determinarea tabelelor din seturile de rezultate MySQL, 283
 obținerea indicatoarelor MySQL, 282
 obținerea lungimii coloanelor, 281
 obținerea structurii complete, 283
 obținerea tipurilor MySQL, 282
 returnate de interogările SELECT, 277
 SGBD (sistem de gestiune a bazelor de date), 360-365. *vezi* și baze de date relaționale
 determinarea bazelor de date găzduite de serverul MySQL, 286
 determinarea coloanelor incluse în cadrul tabelului, 288

determinarea tabelelor incluse într-o bază de date, 287
 tipuri populare de, 236
 shell, definiție, 368
 short_tags, opțiune de configurare, 64
 sigur, argument, 165
 simbol pentru, 367
 simbolul dolarului (\$)
 formarea numelor de variabilă PHP, 28
 secvența escape PHP, 27-28
 sesizarea diferenței între constante și variabile, 71
 simbolul procentului (%)
 utilizarea ca operator modulo, 34
 utilizarea ca procent, în loc de directivă, 144
 utilizarea pentru directive, 142
 Simple Mail Transfer Protocol (SMTP), 209
 sintaxă. *vezi* și mesaje de eroare
 caracteristică de colorare, 15
 definiție, 17
 depanarea erorilor, 23
 mesaje de eroare, 329
 sistem de gestiune a bazelor de date. *vezi*
 SGBD (sistem de gestiune a bazelor de date)
 sistem, administrator de
 cont special de utilizator pentru, 367
 definirea proprietății asupra fișierelor, 173
 probleme de proprietate și privilegii, 181
 situri Web
 consultarea documentației UNIX pe suport electronic, 377
 criptarea datelor stocate în variabile cookie cu Mcrypt, 165
 funcții șir, 148
 funcții tablou, 133
 informații despre XML, 297
 instalarea ODBC, 295
 instalarea și configurarea PHP, 359
 instrumente pentru scrierea programelor PHP, 15
 instrumente și facilități acceptate de PHP, 296
 înțelegerea expresiilor regulate PHP, 153-154

lucrul cu șabloane pentru, 323
 resurse PHP, 362
 SIZE, atribut
 crearea unor casete cu text personalizate, 45-46
 SMTP (Simple Mail Transfer Protocol), 209
 software
 evaluarea performanțelor, 328
 inspecție, 332
 sortare
 datele dintr-o bază de date, 258
 valori calculate sau valori din coloane, 260
 sortare, funcții de ~ pentru tablouri, 134
 sortare, secvență de ~, 134
 specificatori de aliniere, 143
 specificatori de completare, 143
 specificatori de lățime, definiție, 143
 specificatori de tip, tabel cu, 143
 specificatori, șiruri de formatare, 143, 144
 sprintf(), funcție
 crearea datelor de ieșire formate, 142
 inserția simbolurilor procent, 144
 SQL (Structured Query Language), 237
 SQL, interogări
 agregare, 258
 expresii și funcții, 260
 grupare, 259
 modificarea datelor, 255
 proiect, crearea și întreținerea, 265
 sortare, 258, 260
 trecere în revistă, 252
 uniri, 263
 SSH (secure shell service)
 deschiderea sesiunii de lucru la un sistem UNIX, 369
 încărcarea scripturilor PHP, 21
 stabilire, definirea metodelor, 307
 statice (locale), variabile, 114
 stil_etichetă, argument, 292
 stocarea datelor
 scrierea numerelor, 26
 scrierea șirurilor, 27-28
 stop(), funcție, 109
 strcasecmp(), funcție, 148
 strchr(), funcție, 149
 strcmpcase(), funcție, 135

stripslashes(), funcție, 164, 291
 stripslashes(), funcție, 148
 strlen(), funcție, 129, 146
 strncasecmp(), funcție, 148
 strncmp(), funcție, 148
 strpos(), funcție, 148
 strchr(), funcție, 148
 strrchr(), funcție, 148
 strtolower(), funcție, 147
 strtoupper(), funcție, 147
 structură, 234
 str_replace(), funcție, 151
 sub-șiruri
 funcții de căutare, 149
 funcții de extragere, 149
 funcții de înlocuire, 151
 proiect, program de stabilire a
 echivalențelor cu expresii regulate, 155
 subcataloage
 definiție, 365
 formarea căilor relative, 178
 vizualizarea conținutului cataloagelor, 373
 substr(), funcții, 148
 substr_replace(), funcție, 151
 superutilizator, definirea conturilor de, 176
 suprimare, utilizarea funcției
 error_reporting(), 272, 335
 switch, scrierea instrucțiunilor, 93
 switch_to_folder(); funcție, 216
 șabloane, 318
 afișarea variabilelor care conțin rezultatul,
 322-323
 analiza variabilelor asociate fișierelor,
 322-323
 asocierea variabilelor cu fișiere, 321
 atribuirea de valori în, 321
 construcția unui sit Web complet, 322
 crearea fișierelor, 320
 instanțierea obiectului FastTemplate, 320, 322
 introducere, 318
 proiect, lucrul cu, 323
 test de evaluare, întrebări, 326
 test de evaluare, răspunsuri, 355
 șablon, variabile
 afișare, 322-323

analiză, 322-323
 asocierea cu fișierele șablon, 321
 atribuirea unor valori către, 321
 identificare, 320
 șiruri, 139
 șiruri cu ghilimele simple, creare, 141
 șiruri, comparare/căutare, 148
 echivalența caracterelor cu expresii
 regulate, 152
 funcții de comparare, 148
 găsirea și extragerea sub-șirurilor, 149
 înlocuirea sub-șirurilor, 151
 șiruri, creare/afișare, 139
 cu ghilimele simple, 141
 date de ieșire formate, 143
 lucrul cu coduri ASCII, 141
 secvențe escape suplimentare, 139
 trecerea în revistă a elementelor
 fundamentale, 139
 șiruri, manipulare, 146
 amputare, 146
 baleierea datelor de ieșire formate, 154
 conversia la majuscule sau minuscule, 147
 conversie automată de tip, 74
 expresii, 64
 funcții, 416, 261
 funcții suplimentare pentru, 148
 obținerea lungimii, 146
 operator de concatenare, 34
 proiect, generator de știri de senzație, 78
 proiect, program de stabilire a
 echivalențelor cu expresii regulate, 155
 scriere, 27-28
 scrierea de instrucțiuni switch, 94
 test de evaluare, întrebări, 158
 test de evaluare, răspunsuri, 350
 valori, 86

T

tabele
 adăugarea mai multor ~ dintr-o bază de
 date, 257
 determinarea coloanelor incluse în, 288
 determinarea ~lor din interiorul unei baze
 de date, 287

structura unei baze de date relaționale și,
 234
 ștergere, 249
 utilizarea coloanelor cu auto-incrementare,
 275-276
 verificarea interogărilor care nu returnează
 rânduri, 271
 tabele cu detalii, 263
 tablouri, 123
 creare prin atribuire, 123
 creare, utilizarea funcției array(), 125
 funcții pentru, 395
 lucrul cu funcții listă, 131
 multidimensionale, 125
 obiect, 312
 parcursarea prin iterație a ~lor, 126
 parcursarea prin iterație a ~lor
 multidimensionale, 130
 parcursarea prin iterație a ~lor
 non-secvențiale, 129
 parcursarea prin iterație a ~lor secvențiale,
 127
 parcursarea prin iterație a ~lor, instrucțiuni
 break, 128
 parcursarea prin iterație a ~lor, instrucțiuni
 continue, 129
 parcursarea prin iterație a ~lor, vedere de
 ansamblu, 126
 proiect, formular de contact, 136
 sortare, 134
 teste de evaluare, întrebări, 138
 teste de evaluare, răspunsuri, 350
 valori multiple în, 31-32
 tablouri asociative, 124
 tablouri indexate cu șiruri, 124
 tablouri multi-dimensionale
 parcursarea iterativă a, 130
 trecere în revistă, 125
 tablouri non-secvențiale, parcursare prin
 iterație, 129
 tar, comandă, 381
 tarballs, 381
 Telnet, 369
 ternar sau semnul întrebării-două puncte,
 operator (?), 94

test de evaluare, întrebări, 344
 test de evaluare, răspunsuri, 355
 test, expresie
 scrierea instrucțiunilor for, 95-96
 scrierea instrucțiunilor while și do while și,
 97
 teste de evaluare, întrebări
 accesul la bazele de date relaționale, 301
 baze de date relaționale, 268
 crearea formularelor HTML, 60
 crearea programelor PHP, 24
 depanarea scripturilor PHP, 355
 elemente constructive ale limbajului PHP,
 37
 elemente introductive despre bazele de
 date și SQL, 353
 expedierea și recepționarea mesajelor de
 poștă electronică, 232
 funcții, 121
 lucrul cu fișiere și cataloage, 208
 lucrul cu valori scalare, 83
 scrierea instrucțiunilor condiționale, 104
 utilizarea claselor și a obiectelor, 317
 utilizarea șabloanelor de aplicație, 263
 utilizarea șirurilor, 158
 teste de evaluare, răspunsuri
 accesul la bazele de date relaționale, 353
 accesul la date, 347
 crearea formularelor HTML, 346
 crearea programelor PHP, 345
 depanarea scripturilor PHP, 355
 elemente constructive ale limbajului PHP, 346
 elemente introductive despre bazele de
 date și SQL, 353
 expedierea și recepționarea mesajelor de
 poștă electronică, 352
 lucrul cu fișiere și cataloage, 351
 lucrul cu valori scalare, 348
 scrierea instrucțiunilor condiționale, 348
 utilizarea claselor și a obiectelor, 354
 utilizarea funcțiilor, 349
 utilizarea șabloanelor de aplicație, 355
 utilizarea șirurilor, 350
 utilizarea tablourilor, 350
 utilizarea variabilelor cookie, 351

text
citire, 188
conversie în sens escape, 291
TEXTAREA, etichete, 46
time(), funcții
specificarea expirării variabilelor cookie, 161
ștergerea variabilelor cookie, 162
tabel cu funcții MySQL, 262
tabel cu funcții PHP, 399
tip de pericol, 389
tipuri
clase ca fiind definite de utilizator, 303
de baze de date relaționale, 239
de variabile, 29-30
evitarea erorilor la rulare, 331
în sistemul de fișiere UNIX, 173
prototipuri de funcții, 394
vizualizarea informațiilor privind fișierul, 170
tipuri de date, MySQL, 247
tipuri dinamice
conversie automată, 74
conversie manuală, 75
definiție, 74
tranzacții, repudiarea, 390
trim(), funcție, 210
TYPE, atribut
crearea casetelor cu parolă, 46
crearea casetelor de validare, 47
crearea câmpurilor ascunse, 48
expedierea la server a conținutului serverelor, 49

U

uasort(), funcție, 135
uksort(), funcție, 135
unire exterioară la stânga, 264
uniri
accesul la datele din baze de date SQL, 263
exterioare la stânga, 264
trecere în revistă, 263
UNIX System Security Tools (Ross), 393

UNIX, 469-475. *vezi și* cataloage, UNIX;
fișiere, UNIX
cataloage, 365
căi, 365
conturi de utilizator, 367
editoare de text pentru scrierea scripturilor PHP, 14
fișiere, 365
grupuri de utilizatori, 367
instalarea PHP, 359
încărcarea scripturilor PHP prin intermediul, 19-21
proprietate și permisiuni, 368
trecere în revistă, 364
UNIX, tehnici, 369
citirea și expedierea mesajelor de e-mail, 372
compararea fișierelor text, 380
consultarea documentației UNIX pe suport electronic, 377
copierea fișierelor sau a cataloagelor, 375-376
crearea cataloagelor, 375
deschiderea și închiderea sesiunii de lucru, 369
determinarea tipului fișierului, 380
determinarea utilizatorilor conectați, 371-372
editarea fișierelor, 375
emiterea comenzilor, 370
găsirea fișierelor, 379
lucrul cu fișiere compresate, 380
lucrul cu fișiere tar, 381
metacaractere de shell și globalizarea numelor, 382
modificarea catalogului curent de lucru, 479-480
modificarea numelor fișierelor sau a cataloagelor, 376
modificarea parolei, 371
raportarea gradului de utilizare a spațiului pe disc, 377
stabilirea permisiunilor pentru fișiere, 378
stabilirea proprietății asupra unui fișier, 378
ștergerea cataloagelor, 375-376

ștergerea fișierelor, 375
trecere în revistă, 369
verificarea gazdei, 371-372
vizualizarea conținutului cataloagelor, 373
vizualizarea conținutului fișierelor și a datelor de ieșire ale comenzilor, 374
unlink(), funcție, 199
unserialize(), funcție, 163
UPDATE, comandă, 256
UPDATE, interogări, 271
update_database(), funcție, 120
urdecode(), funcție, 292
usort(), funcție, 135
utilizarea procedurii cu tabloul de obiecte, 314
utilizatori
autentificare și autorizare, 391
suspectarea datelor de intrare provenite de la, 391
utilizatori, argument, 176

V

valoare, 161
valori
atribuirea de ~ variabilelor șablon, 321
definirea și utilizarea ~lor true sau false, 86
evitarea erorilor la rulare, 331
modificarea ~lor prin referințe, 114
returnate de funcții, 105
stocate în variabile cookie, 163
tabele din bazele de date, 244
valori numerice
care formează expresiile condiționale, comparație, 86
proiect, testare, 89
valori returnate, 105
valori true
definire și utilizare, 86
scrierea de instrucțiuni if complexe, 91
scrierea de instrucțiuni if simple, 89
utilizarea funcției ereg(), 153-154
VALUE, atribut
crearea casetelor cu text personalizate, 45-46
crearea casetelor de validare, 47
crearea câmpurilor ascunse, 48
crearea selecțiilor, 47
expedierea către servere a conținutului fișierelor, 48
încorporarea controalelor, 41
proiect, program de navigare în agenda cu adrese, 205
var, instrucțiuni, 306
variabile cookie, 159
acces la, 160
antete HTTP și, 161
creare, 161
criptarea datelor stocate în, 165
limitări de stocare pentru, 159
proiect: construcția unei pagini de deschidere a sesiunii de lucru, 166
specificarea accesului la, 164
stocarea a mai mult de 4KB de date, 164
stocarea mai multor valori în, 163
ștergere, 162
test de evaluare, întrebări, 168
test de evaluare, răspunsuri, 351
utilizare, 159
variabile de mediu, 67
definiție, 67
proiect, vizualizare, 69
tabel cu ~ importante, 67
test de evaluare, întrebări, 70
test de evaluare, răspunsuri, 347
trecere în revistă, 67
utilizări ale, 69
variabile dinamice, 72
variabile globale
comparație cu variabilele locale, 113
trecere în revistă, 112
variabile. *vezi și* tablouri; valori scalare
funcții pentru, 420
practici de depanare, 339-340
proiect, vizualizarea valorilor ~lor, 31
trecere în revistă, 28-30
utilizarea ~lor globale, 112
utilizarea ~lor locale și a celor statice, 113
verificarea apariției erorilor
biblioteca MySQL, 271-272
determinarea bazelor de date găzduite de serverul MySQL, 287

determinarea coloanelor incluse într-un tabel, 287
determinarea tabelelor incluse într-o bază de date, 287
virgulă (,)
instrucțiuni echo care folosesc, 64
specificarea argumentelor, 394

W

w, comandă, 371-372
w, privilegiu
privilegiu de catalog, 180
privilegiu de fișier, 175
WHERE, clauză
gruparea bazei de date cu, 258
mai multe tabele într-o singură interogare, 263
sortarea pe valori calculate sau pe valori dintr-o coloană, 259-260

while, instrucțiuni
alegere, 98
scriere, 97
Windows. *vezi* Microsoft Windows
WRAP, atribut, 46

X

x, privilegiu
privilegiu de catalog, 180
privilegiu de fișier, 175
XML (Extensible Markup Language), 295, 297
xpat, bibliotecă, 295

Z

zecimale, argument, 145
zip, comenzi pentru fișierul, 380